CA Endevor® Software Change Manager

Administration Guide

Version 17.0.00

Fourth Edition

CA Technologies Product References

This document references the following CA Technologies products:

- CA Endevor® Software Change Manager (CA Endevor SCM)
- CA Change Manager Enterprise Workbench (CA CMEW)
- CA ACF2™ for z/OS (CA ACF2)
- CA Endevor®/DB for CA IDMS™ (CA Endevor/DB for CA IDMS)
- CA Librarian Base for z/OS® (CA Librarian Base)
- CA Panvalet for z/OS® (CA Panvalet)
- CA Roscoe® Interactive Environment (CA Roscoe IE)
- CA Top Secret ® for z/OS (CA Top Secret)
- CA 7™ Workload Automation (CA 7 Workload Automation)

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

Note: In PDF format, page references identify the first page of the topic in which a change was made. The actual change may appear on a later page.

Version 17.0, Fourth Edition

- Alter Action Updates Master Control File Element Metadata— Updated to add an important note about security.

Version 17.0, Third Edition

- Dialog Options Fields (see page 198)— Updated to remove the obsolete field DISPLAY_PROC_GROUP.

Version 17.0, Second Edition

- Endevor Actions (see page 33)— Updated to add the Alter action and to remove a misplaced statement from the Update action description.
- Using SMF Recording (see page 173)— Updated this chapter to add changes for the Alter action. Also, the DSECT samples have been removed from the guide. All the DSECTs are delivered in the installation CSIQOPTN library.
- Altering Element Record Metadata (see page 255)— Added this chapter to describe the Alter action.

Version 16.0

- Physical Structure (see page 25)— Updated to change the version and level format to VVLL in the graphic.
- Function Controls Fields (see page 76)— Updated to change the description of the MODLHI parameter on the Site Information panel.
- Element Options (see page 103)— Updated to add the ELE RECFM field on the Type Definition panel. This parameter specifies whether the temporary CA Endevor Quick Edit ISPF edit data set's record length format is fixed, variable, or not defined. Also, clarified the description for LIBENV and added a cross-reference in the field description for PV/LB LANG.
- How to Define an Element Type for Text Files (see page 118)— Added to describe this process.
- $SMFBKDS DSECT: Action-Specific Blocks (see page 182)— Updated to add the field SM2RSPAN for the Autogen Span action options.

- Request Parameter Info Action-Block Detail Field Descriptions (SM2REQDS) (see page 186)— Updated to add the field SM2RSPAN for the Autogen Span action options.
- Global Type Sequencing—The chapter "Performance Improving Options" was revised to remove topics about the Global Type Sequencing feature. The content for this feature was restyled into the scenario-based knowledge document *How to Enable Global Type Sequencing*, which can be found in the *Scenario Guide*. This scenario includes changes to the following topic:
  - Type Sequence Record Syntax in Type Sequence File—Updated to add the Category clause to the Type record syntax in the Global Type Sequence member.
- Autogen— The chapter "Performance Improving Options" was revised to remove topics about the Autogen action option. The content for Autogen was restyled into the scenario-based knowledge document *How to Automatically Generate Using Elements*, which can be found in the *Scenario Guide*. This scenario includes changes to the following topics:
  - Autogen Action Option— Updated to add the AUTOGEN_SOURCE option for ENCOPTBL, which changes the behavior of Autogen. Also, added a reference to the Autogen Span options.
  - How Generating Elements with Autogen Works— Updated to add a note about the AUTOGEN_SOURCE option for ENCOPTBL.
  - How Autogen Span Options Work— Added to introduce the Span parameter, which enables Autogen to locate and generate using elements found in different Systems and Subsystems within the Environment and Stage of the target element's logical map.
  - Run Autogen in Simulation Mode for Impact Analysis— Added to describe this procedure.
- Dialog Options Fields (see page 198)— Updated to add the ISPF Dialog field names AUTOGEN_SPAN and QE_AUTOGEN_SPAN. Also, updated to add PRESERVE_VB for CA Endevor Quick Edit ISPF edit sessions.
- Aged Delta Retention (see page 239)— Updated to change the version and level format to VVLL in the examples.
- Source Synch Check Email Format (see page 314)— Updated to change the version and level format to VVLL in an example.
- USS Supported Files and the Alternate ID (see page 307)— Updated to correct references to the Optional Features table (ENCOPTBL) option: ENABLE_ALTID_USS_SECURITY.
- The TYPE=MAIN Macro (see page 208)— Updated to add DESTCFGMBR for the Post-Ship Script feature. Updated to add an additional use for the MODHLI parameter for client programs that accesses the API through Web Services, such as the Eclipse-Based UI, CA CMEW, or a user written client program.

- Appendix, "Working with Binary Files"— This appendix was removed, because its content How to Define an Element Type for Binary Files (see page 120) was edited for clarity and moved.
- How to Enable Trace Facilities (see page 317)— Updated to add the trace EN$TRLOG for logon and logoff information. Also, updated the description for the EN$TRAUI trace to state that it also traces logon and logoff information, in addition to the Alternate ID.

Release 15.1

- Log Delta Format (see page 116)—Updated to add HFS and binary files to the suggested file types for this delta format.
- Performance Improving Options (see page 193)—This chapter was revised to remove topics about the Concurrent Action Processing feature. These topics were moved and restyled into the scenario knowledge document *Concurrent Action Processing.* This document can be accessed from the documentation HTML bookshelf.
- How to Define an Element Type for Binary Files—Updated to recommend the log delta format for binary files.
- USS Supported Files and the Alternate ID (see page 307)—Added to describe when the Alternate ID or the user ID is used to process USS supported files.

Version 15.0

- Stage Information Panel Fields (see page 86)—Added the Stop at Stage fields.
- Element Registration Options (see page 92)—Updated to add the ACROSS SBS field for the duplicate processor output Type check.
- Subsystem Definition Panel Fields (see page 99)—Updated to add the option to exclude the Subsystem from the processor output Type check.
- Defining Types
  - Element Options (see page 103)—Updated the field FWD/REV/IMG/LOG DELTA to include the log delta option.
- Element Storage Formats (see page 113)—Updated to add the Log delta format.
- Log Delta Format (see page 116)—Added to describe the Log delta format.
- Controlling Duplicate Element Names at the Processor Group Level (see page 139)—Updated to add the option to extend the processor output type check to the Subsystem level.
- How to Enable Element Registration for Processor Groups (see page 139)—Added to describe the steps to set up Element Registration for processor groups.
- Add the Data Set Name to the Defaults Table (see page 143)—Updated the parameter example to show an equal sign (=) instead of a dash (-) in the parm=parmvalue.

- Specify TYPE=MAIN Parameters—Updated the parameter example to show an equal sign (=) instead of a dash (-) in the parm=parmvalue.
- Specify TYPE=ENVRNMNT Parameters—Updated the parameter example to show an equal sign (=) instead of a dash (-) in the parm=parmvalue.
- Set the CA Panvalet Parameters in the Defaults Table (see page 145)—Updated the parameter example to show an equal sign (=) instead of a dash (-) in the parm=parmvalue.
- Set the CA Librarian Parameters in the Defaults Table (see page 146)—Updated the parameter example to show an equal sign (=) instead of a dash (-) in the parm=parmvalue.
- Restore Action and Converting Delta Formats (see page 273)—Added to describe this concept.
- How to Convert Elements to Log Delta Format (see page 273)—Added to describe the procedure.

Contents

Chapter 1: Software Development Change Control

This section contains the following topics:

Overview

CA Endevor SCM is an integrated set of management tools used to automate, control, and monitor your application development process. You can use this software to perform the following activities:

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

- Put change packages and approvals online, eliminating change-related paperwork.
- View or retrieve prior levels of any element.
- Report on element definition, content, and change history.
- Enforce change control procedures.

CA Endevor SCM is implemented and run under z/OS, within the TSO ISPF environment, and in batch. Your site can also run an API program that interfaces with CA Endevor SCM.

Audience

This guide address the tasks of the CA Endevor SCM administrator. The following tasks are discussed in this guide.

- define the software management lifecycle in CA Endevor SCM
- define and place the inventory of software components under the control of CA Endevor SCM
- define and maintain maps that control how elements progress through the lifecycle
- define and maintain global options

What You Need to Know

To use the product, you need a working knowledge of the mainframe environment, the z/OS mainframe operating system, Time Sharing Option facility (TSO), and the Interactive System Productivity Facility (ISPF). It is also assumed that the product has been properly installed at your site.

This guide describes the tasks that administrators can perform in the ISPF environment. Foreground and many batch tasks can be performed using the ISPF panel driven interface. Batch JCL streams can also be written using the standard ISPF editor and submitted from TSO/ISPF.

For more information about writing batch JCL streams, see the *SCL Reference Guide* and the *Packages Guide*.

For information about installing and configuring the product, see the *Installation Guide*.

For more information about terms used in this guide, see the "Glossary" in the *User Guide*.

The Software Lifecycle

CA Endevor SCM allows you to automate and control the movement of software through your software lifecycle.

Software lifecycles are site-specific. A representative lifecycle might consist of five stages:

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

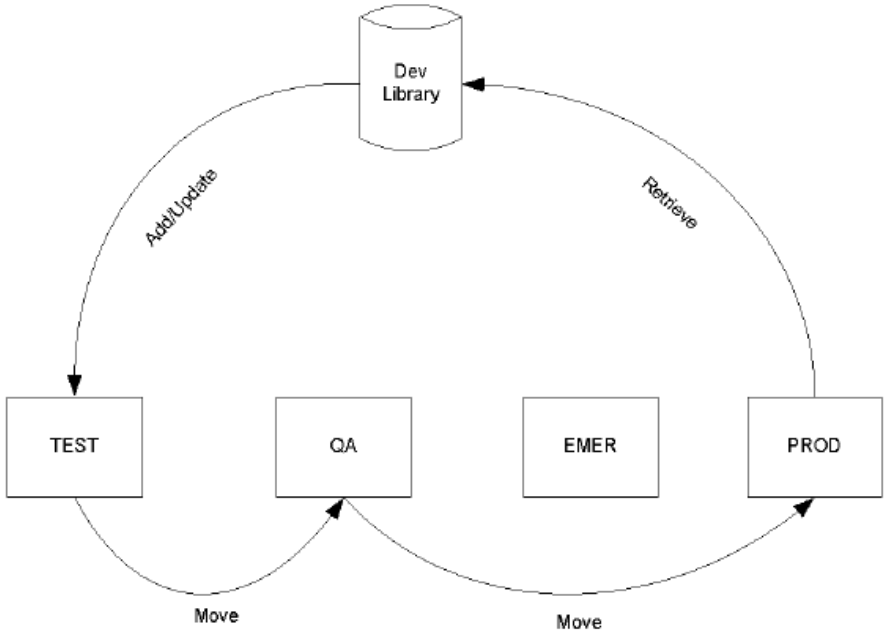
Note: This example illustrates one lifecycle. CA Endevor SCM can be implemented to adapt to any software lifecycle requirements.

Basic Operations

Normal change procedures include the following:

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

The following diagram shows normal change procedures in a software lifecycle:

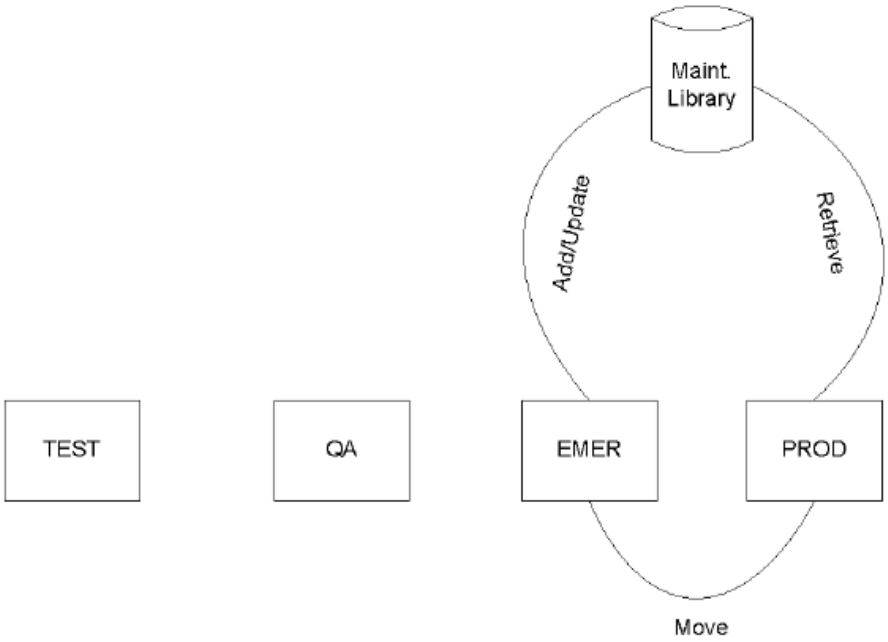


Emergency Operations

Emergency change procedures include the following:

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

The following diagram illustrates emergency change procedures in a software lifecycle:



Conceptual Overview

CA Endevor SCM provides automated facilities for performing all software management tasks from inventory management and change control to configuration and release management. These facilities are further enhanced through comprehensive change administration, parallel development management, software security and software information management facilities. As a fully-integrated, single-vendor solution, CA Endevor SCM dramatically improves the operation and administration of IBM mainframe installations by providing the following:

- A comprehensive inventory of all programs and software assets that reside in partitioned data sets (PDSs), library management systems, HFS directories, and executable libraries
- An absolute history of all changes that have occurred to the source
- Control of the processes and procedures that translate source into executable forms
- An inviolate and auditable link between the source code and its related executable forms
- Protection and control of inventory items through extended security

- Automated cross-referencing of software component relationships for purposes of historical analysis, change impact analysis, recreation of prior versions, and release management
- Control and automation of the movement and distribution of software release packages from stage to stage, site to site, and across networks

CA Endevor SCM accommodates the diversity of both small-scale and large-scale IS operations. And CA Endevor SCM works in conjunction with existing procedures, structures and standards, rather than imposing new ones.

Inventory Management

In most organizations, the application inventory is large, complex, and always changing. Today's applications have graduated beyond standard source to include a wide variety of components—most outstripping traditional 80-character storage limitations.

In addition to the changing complexion of the inventory's physical structure, there is a growing proliferation of physical libraries and functional environments (Test, QA, Production, and so on). Programmers and IS support personnel typically require intimate knowledge of those library structures as they relate logically to business units within an organization.

This logical view, divorced from the physical storage structure, is required both to ensure that logically related inventory elements are being manipulated consistently and correctly, and to provide a common user interface regardless of physical structures. An effective inventory management structure also forms the foundation for change control, configuration management, and release management activities.

Automated Inventory Manager

Traditional software classification systems (PDSs, library management systems, and so forth) provide limited inventory classification capabilities and highly restrictive methods for storing and recreating prior versions of software modules.

When using these traditional systems, IS departments resort to complex naming conventions and physical separate libraries to obtain a semblance of organization. This indirect naming and storage technique is both error-prone and inflexible.

Consequently, as programmers move from project team to project team, they are forced to learn and relearn the peculiarities of manipulating the software inventory as applied by each project team. Clearly, a common technique is required for classifying software throughout the IS organization, one that provides both consistency and flexibility.

Logical Structure

CA Endevor SCM employs advanced classification techniques that form the necessary foundation for categorizing, viewing, and manipulating the application software inventory in a consistent manner. An inventory item (element) is identified to CA Endevor SCM's Inventory manager by a fully-qualified name consisting of its environment, stage (location), system, subsystem, type, and element name.

Environment, Stage

As an application is modified and its elements are moved throughout the software development lifecycle, those elements may reside in different functional locations (Test, QA, Production, Backup, and so on) at different times. Within CA Endevor SCM's inventory classification scheme, the location of an element is part of the identification of that element.

System

Systems represent the logical grouping of inventory elements as they apply to major applications, departments, or work areas within an organization. Typically, a system is the "organizational owner" of a portion of the inventory. For example, the systems in an organization might include Finance, Personnel, and Manufacturing.

Subsystem

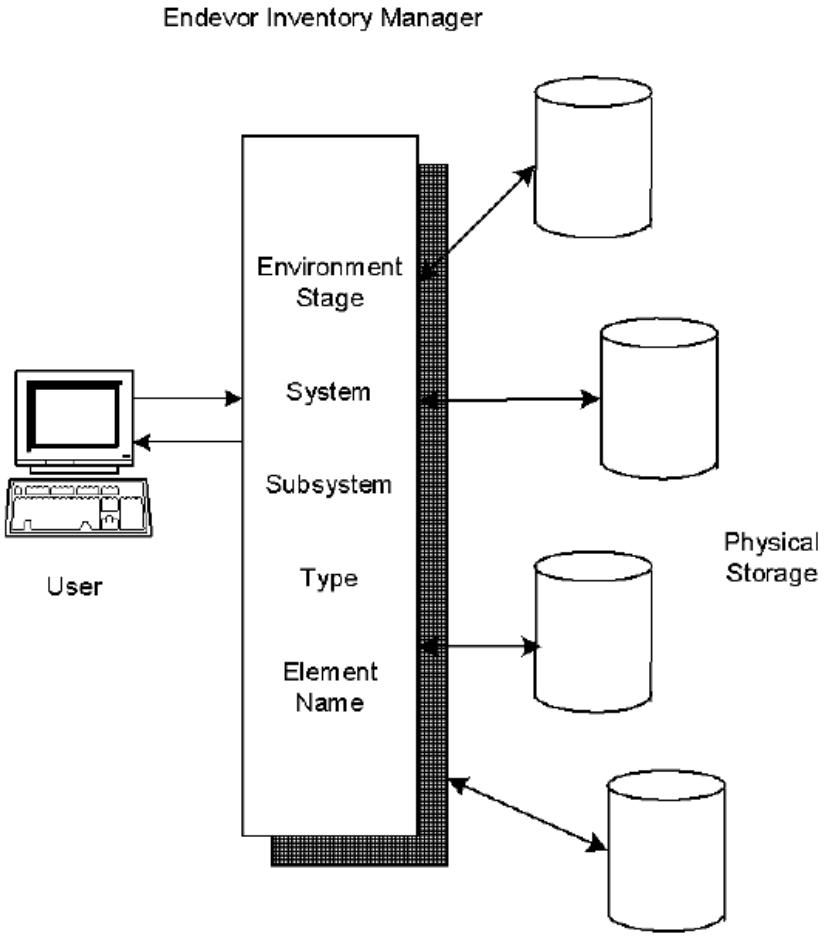
Subsystems are logical sub-classifications within systems. For example, the Finance system might be divided into logical subsystems such as General Ledger, Accounts Receivable, Accounts Payable, Common Routines, and Reports.

Type

Within the application inventory, types represent the form of the element, indicating how the element is created (the source language used) and how it is manipulated. For example, the Finance system might have several types of elements, including COBOL programs, Assembler programs, C programs, PL/I programs, copy members, Assembler macros, screen definitions, and linkage editor statements.

Element Name

The element name is the existing name of the element because it is already established. This logical classification scheme, as illustrated in the following diagram, provides a consistent view of the inventory without requiring the user to understand the inner workings of the inventory's physical structure.



In CA Endevor SCM, the logical attributes, which are used to classify an element, include its location, system, subsystem, and type. These attributes are a direct part of the identification of that element. This eliminates the need to establish additional libraries, rename elements, or use cryptic naming conventions to define and maintain a classification scheme. Because these classification attributes are used in addition to the existing element name, there is no need to rename elements to bring them under the control of the CA Endevor SCM Inventory Manager.

Physical Structure

In today's IT environment, the physical requirements of a properly managed inventory have grown beyond the capabilities of older library management systems and conventional PDS-based systems. Many library management systems are unable to store source that is longer than 80 characters, despite the fact that many modern-day languages do not conform to this format.

Conventional PDSs do not restrict record length, but elements of different record lengths must reside in physically separate libraries. Using these structures, the only way to keep multiple revisions of an element is to either replicate libraries to hold prior revisions or use a naming convention in which the revision is part of the element name.

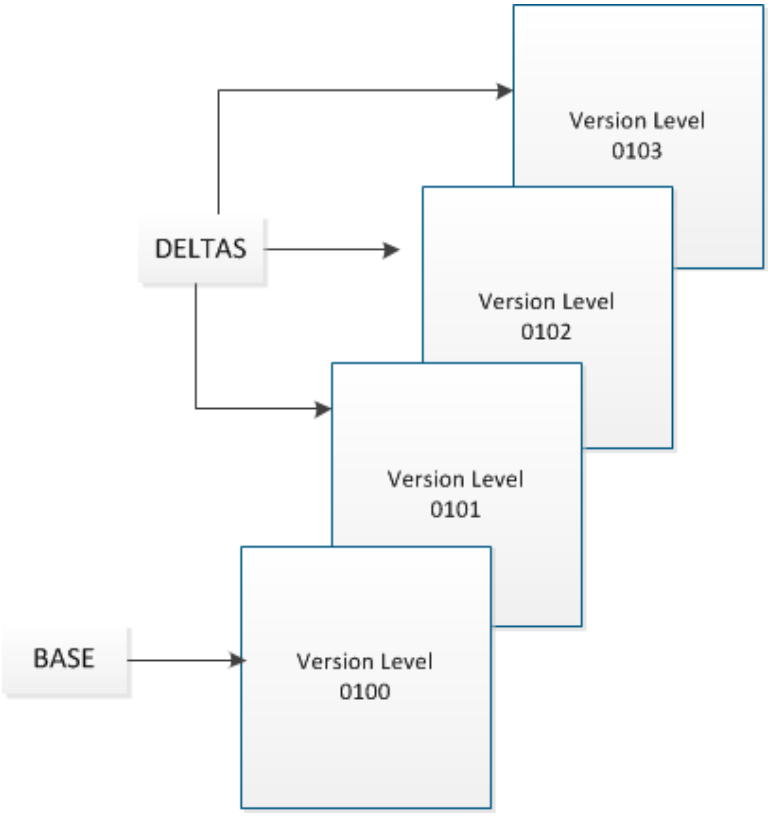
Physical repository

CA Endevor SCM's Inventory Manager has the flexibility to manage the physical data sets in which elements are stored, without regard to logical classification boundaries or trivial physical differences between the inventory elements. It can do the following:

- Handle unrestricted source record lengths
- Store elements of different record lengths in the same repository
- Store multiple revisions of an element in a single repository
- Store elements of the same name in the same repository, regardless of differences in logical classification
- Change the underlying physical library structure without affecting the logical view of the inventory

Base and Delta

CA Endevor SCM employs advanced base and delta technology to store the elements within the application inventory. The first time an element is added into CA Endevor SCM, it is stored in its entirety as the base. Subsequent updates to that element create records containing only the changes to the element. These deltas are automatically assigned level numbers and stored by CA Endevor SCM, as shown in the following illustration:



This method yields substantial savings in storage overhead by providing an efficient means of storing multiple revisions of an element. It also provides a wide variety of ways to view change information, including current, historical or changes only, available online or in hardcopy format. All change displays identify what changes were made and by whom, when and why. This is known as the forward delta format. Reverse delta format processing is similar; however, the delta source is kept in the current format and deltas are used to recreate prior iterations of the source.

Interfaces to CA Panvalet and CA Librarian

Interfaces are available that allow CA Endevor SCM to read directly from and write directly to CA Panvalet and CA Librarian files. Additionally, these interfaces allow you to use CA Panvalet and CA Librarian files to contain base/delta source within CA Endevor SCM. In this way, your initial investment in library management systems is enhanced rather than rendered obsolete.

Change Control

Applications typically undergo extensive modifications as companies customize and fine-tune software systems to meet their specific business requirements. Given the sheer volume of change requests in today's IT organization and the growing demand for more effective ways of tracking and documenting development change activity, manual change control procedures can no longer be relied upon to produce the accurate, reliable information needed to identify problems, track projects, and analyze the impact of change.

Automated Change Control

CA Endevor SCM forms an envelope around your application environment and captures all change events, enabling you to identify what was changed, who made the changes, when, and why. And since CA Endevor SCM performs change control functions automatically, the resulting information is always accurate and up-to-date.

Identifying and Tracking Changes

As a prerequisite to understanding, controlling, and auditing the software environment, change control systems must be able to identify changes as they occur, then further pinpoint when the changes were made, by whom, when, and why.

CCIDs

CA Endevor SCM Change Control Identifiers (CCIDs) provide a means of grouping and manipulating similar kinds of change activity. For example, a number of different elements (COBOL copy members, COBOL programs, and so forth) can be included in a single change request. By tagging each of those changes with a common CCID (such as FIX01), every change made for that change request is categorized as belonging to that CCID. Subsequently, the CCID can be used to identify and manipulate all the elements within the change request. For example: Move all elements for COD FIX01 from Unit Test to System Test. CCIDs can also be predefined and validated for change administration purposes.

Change History

CA Endevor SCM automatically captures and dynamically tracks all source changes by creating a unique delta to record each change event. An element's delta levels identify not only the actual lines of code that have been changed, but also user, comment, CCID, and date and time information for that change. The result is a complete and accurate history or audit trail of application change activity.

Regression Notification

In parallel development situations where many individuals work on the same module concurrently, change regression occurs when one individual unintentionally negates (rather than incorporates) the changes made by another individual. When unchecked, regression can be a major cause of production failures. During action processing, CA Endevor SCM performs a regression check and notifies you when regression has occurred, allowing you to quickly pinpoint and correct the situation before it adversely affects the production environment.

Signin, Signout

In order to avoid possible regression during parallel development, CA Endevor SCM provides signin/signout security at the element level. This optional capability is enabled on a system-by-system basis. When the facility is enabled, elements are signed out as they are retrieved. While other users can retrieve copies of those signed out elements, they cannot update those copies within CA Endevor SCM without explicitly overriding the original user's signout. Only authorized users can exercise the explicit override feature.

Activity Logging

A complete log of all activity within CA Endevor SCM can be written to a variety of data structures, including standard SMF records. This information, available online and in hardcopy format, provides a high level audit trail of change events which can be used to determine change patterns over time.

InfoMan

CA Endevor SCM provides a flexible, programmable interface to the IBM InfoMan change management product. This interface allows certain change administration functions to be performed through InfoMan on behalf of CA Endevor SCM.

These functions include: the validation of CCIDs against the InfoMan database; the logging of CA Endevor SCM actions to InfoMan; the ability to use the InfoMan approval and notification facilities; the ability to control CA Endevor SCM actions based on InfoMan CCID status information; and the ability to query and utilize InfoMan change, problem, and configuration information during CA Endevor SCM action processing.

Automating Processes

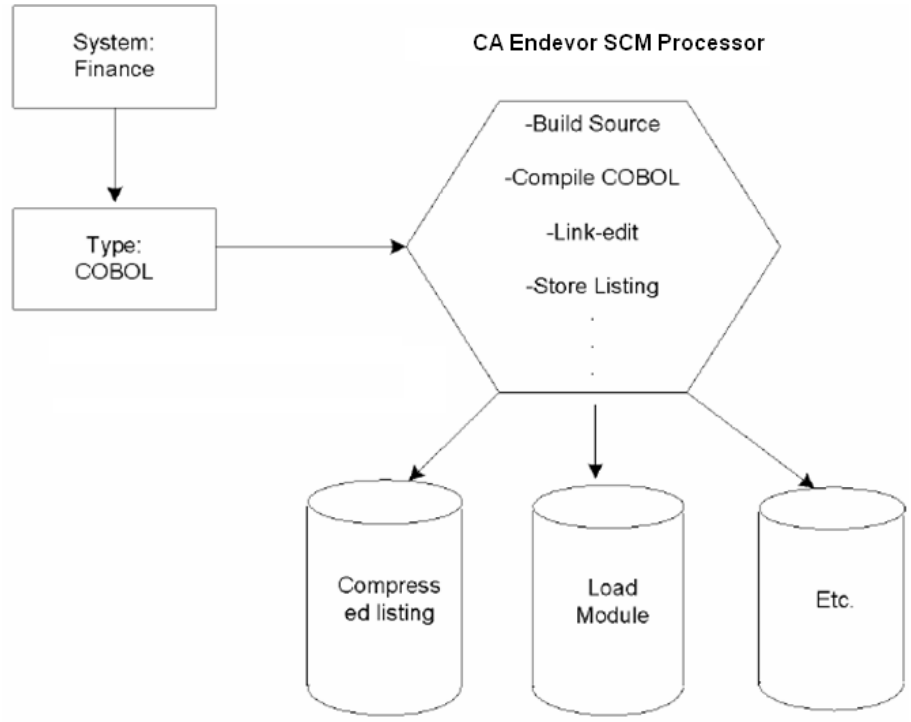
In application development, a procedure or process is the set of steps used to transform source language statements into executable code. Some inventory elements, such as those written in 3rd generation languages (COBOL, Assembler, C, PL/I, and so on) have straightforward processes that compile the source and link-edit the object. Other element types require more complex processes, such as the resolution of data base calls or the automated generation of textual documents. Because these processes ultimately determine how an element will interact in production, the management of these procedures is critical.

One of the powerful facilities provided by CA Endevor SCM involves the regulation and control of all transformation processes through automated procedures called processors. CA Endevor SCM processors are written in job control language (JCL) statements that specify process steps, parameters, error conditions, data set names, output libraries and the like.

Once defined, processors can be used to automatically carry out a variety of procedures and create the outputs associated with application elements. Element types within the inventory classification determine the appropriate processors. For instance, elements that are written in COBOL are typically handled by the COBOL compiler and linkage editor, while elements that are classified as copy members are usually validated and copied to a specific location.

Automated Output Generation

CA Endevor SCM processors are often used to automate the procedures that generate outputs from source. As illustrated in the following diagram, you can create a processor for Batch COBOL programs that automatically performs a COBOL compile and link-edit into the Finance system's load library, and stores the listing in the Finance system's listing library.



Automated Promotion Procedures

A processor can also contain procedures for promoting applications throughout the development lifecycle. When CA Endevor SCM is instructed to perform a promotion, it reads the processor definition to obtain directions about the promotion procedures specified for a selected element type. Those procedures are then uniformly invoked by CA Endevor SCM. When promoting a Batch COBOL program within the Finance system from Testing to Production, for example, a MOVE processor can be defined which contains instructions for automatically copying the load modules from the Test load library to the Production load library.

Automated Backup and Recovery

In addition to automating the procedures used to generate outputs from source and promote applications throughout the development lifecycle, processors can be defined to execute emergency backup and recovery procedures. For example, a MOVE processor can contain a step that automatically copies production load modules to a backup library. A recovery procedure can additionally be defined to copy that module from the backup library into an emergency library in the event of a production failure.

How to Establish Source-to-Executable Synchronization

In order to effectively and accurately maintain, debug, and audit software, you must be able to link executable code back to its originating source. CA Endevor SCM achieves this goal by controlling the processes by which outputs are created from the source, and further ensures source-to-executable synchronization by audit stamping or footprinting those outputs.

Process Control

CA Endevor SCM's automated processors ensure that the source-to-executable link is not compromised, by tying the process used to create outputs directly to the manipulation of the source. Frequently, load modules and their associated source are processed at different points in time, presenting a precarious window in which a change to either one causes an out-of-sync condition. When source is presented to CA Endevor SCM, the outputs are automatically created from the current source through the use of automated processors. When source is moved through the development lifecycle, CA Endevor SCM's automated processors ensure that the load modules and outputs are manipulated together with the source, rather than through separate procedures.

Footprints

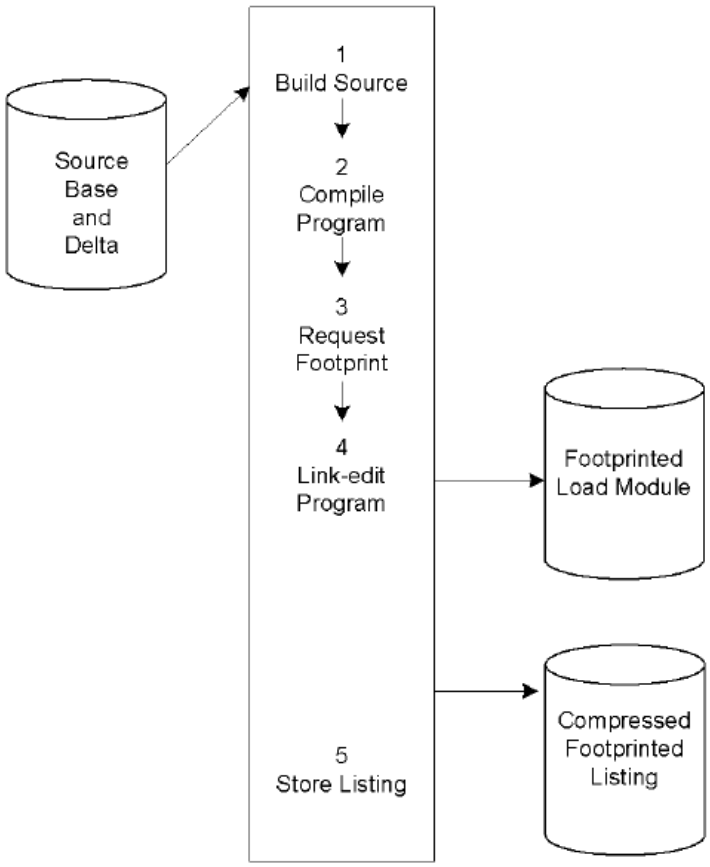
Footprinting is a change control technique employed by CA Endevor SCM to maintain synchronization between source and its related executable. During element processing, CA Endevor SCM places an encrypted audit stamp called a footprint in the output source, object, or load modules that are created. Only CA Endevor SCM automatically builds footprints as an integral part of the source-to-executable transformation process.

CA Endevor SCM footprints contain the following information:

- Location information
- Element name
- Element type
- Element version/level
- System and subsystem to which the element belongs
- Date/time the footprint was assigned

CA Endevor SCM employs standard operating system facilities to store its footprints, thereby ensuring compatibility with present and future operating systems. In a PDS, for example, the footprint is stored in the user data area of the directory; in CA Librarian, as a history record; in CA Panvalet, as a comment field; in a load module, in the user IDR record for the CSECT.

All data structures managed by CA Endevor SCM (source libraries, copy libraries, executable libraries, and processor listing libraries) contain footprinted elements. Footprints can be used to validate the source-to-executable link of an entire load library. And because CA Endevor SCM footprints uniquely identify the versions/levels of input components, they play a key role in configuration management. The following diagram provides a high-level view of how automated processors and footprints work together:



As shown in the previous illustration:

1. The source is built from CA Endevor SCM's internal base/delta format, and a footprint representing that element's source is created and made available to subsequent steps in the CA Endevor SCM processor; the source is compiled.
2. The output object from the compile is stamped with a footprint before being passed to the next step.
3. The linkage editor links the program into the appropriate load library.
4. The listings from the previous steps are combined and stored (also with a footprint) in the appropriate listing library.

Audit Management

As valuable corporate assets, applications cannot be compromised when it comes to integrity, reliability, and recoverability. For this reason, applications must be controlled and secured in a way that meets stringent auditability requirements. CA Endevor SCM's automated change control functionality fully satisfies the scope of an audit and provides the necessary platform for producing management reports that can be used to evaluate software integrity.

Endevor Actions

CA Endevor SCM performs software management operations through user-invoked actions. All actions can run in both foreground (online) and background (batch) modes, except for the Archive, Alter, Restore, Transfer, and Validate actions. The actions are as follows:

ADD

The action used to introduce an Element into a CA Endevor SCM environment.

ALTER

This action replaces Master Control File Element metadata with user-specified values. The Alter action is not available from the Foreground Options menu. Alter statements can be submitted from the Batch Options menu.

ARCHIVE

The action used to copy an Element and all of its change history and control information from a CA Endevor SCM environment to an external data set. Archive can only be run in batch mode.

DELETE

The action used to remove an Element from a CA Endevor SCM environment.

GENERATE

The action used to execute the CA Endevor SCM automated processor for an Element.

LIST

The action used to scan members in a library or the CA Endevor SCM inventory to generate a list of Elements that meet specific selection criteria.

MOVE

The action used to promote an Element from stage to stage within a CA Endevor SCM environment.

PRINT

The action used to produce hardcopy output of CA Endevor SCM Element, change, change history information, and Element output listing.

RESTORE

The action used to re-introduce an Element to CA Endevor SCM from an archived data set. Restore can only be run in batch mode.

RETRIEVE

The action used to copy any level of the source of an Element from a CA Endevor SCM environment into an external data set.

SIGNIN

The action used to remove the user signout associated with an Element.

TRANSFER

The action used to transport an Element from one location to another, where each location can be either a CA Endevor SCM location or an external data set. Transfer can only be run in batch mode.

UPDATE

The action used to create a revision of an Element in a CA Endevor SCM environment. When applicable, these actions drive the automated processors that have been defined to CA Endevor SCM. Before an action is executed by CA Endevor SCM, a security check confirms the user's authority to perform that action upon the requested Element. All actions and their results are reflected in CA Endevor SCM execution processing reports.

VALIDATE

The *Validate action* performs element master, synchronization, and component validation checks against the element you specify. These inventory validation checks can be performed at any time, enabling developers to ensure the integrity of elements before including them in a package. This is a batch only action and is not supported in a package.

Software Control Language (SCL)

The tedious and repetitive tasks associated with manipulating the software inventory are time-consuming and prone to error. CA Endevor SCM's Software Control Language (SCL) is a powerful yet simple language which, when combined with CA Endevor SCM actions, eases the burden of manipulating and moving portions of the software inventory throughout the entire development lifecycle.

SCL Capabilities

SCL saves substantial amounts of time by enabling you to work with as many (or as few) CA Endevor SCM actions as are required to complete a specific job. With SCL, you can do the following:

- Perform bulk data manipulation.
- Set up a single list or multiple lists of actions for manipulation by CA Endevor SCM.
- Establish standardized global settings for action requests.
- Process actions in inventory type sequence order, automatically sorting elements according to specification.
- Generate software configuration lists based on different selection criteria.
- Use a single scan facility that will run against CA Panvalet, CA Librarian, a PDS, and CA Endevor SCM, eliminating the need to use separate utilities to scan source code.
- For problem-solving purposes, generate a list of only those elements that were not successfully processed at a specific time.
- Write user-defined front-ends for various vendor-supplied programs.
- Integrate CA Endevor SCM into existing job scheduling systems.

SCL can be generated and manipulated through online screens, providing an easy-to-use, flexible environment for executing software procedures. Using SCL, repetitive tasks, which normally require significant amounts of time, can be accomplished with very little effort.

Configuration Management

Historically, a major deficiency in the management of software systems has been the lack of an accurate means of determining the interdependencies or configurations of applications and their related components. Semi-automated source scanning techniques have provided a partial solution. These techniques are no longer feasible, however, given the size and complexity of today's applications, the frequency of component changes, and the growing demand for a more auditable means of tracking and managing software configurations as they evolve over time.

Automated Configuration Option

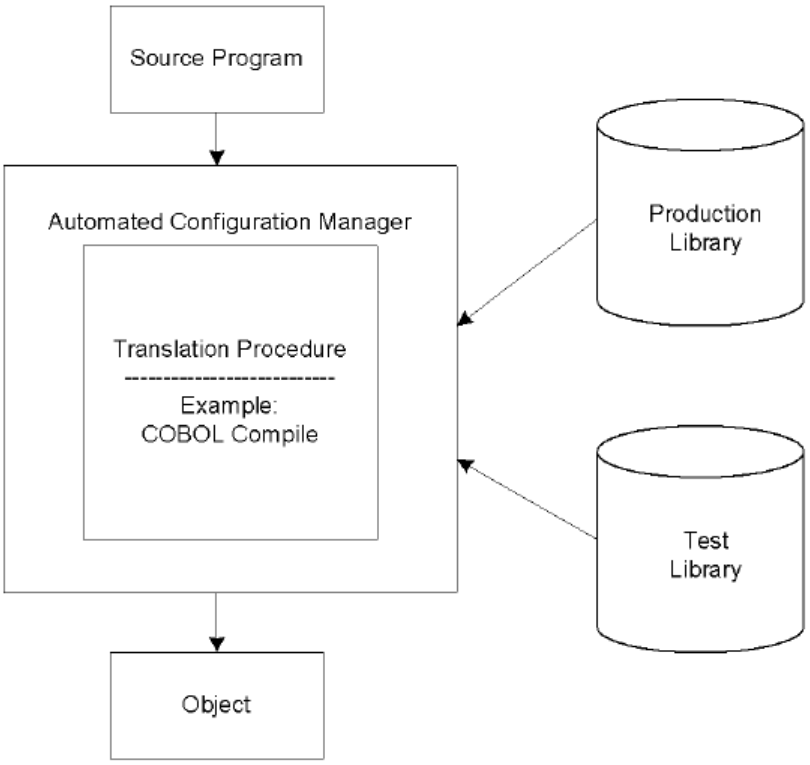
Built upon a logical inventory structure and a powerful change control platform, CA Endevor SCM provides an automated configuration management facility (ACM) that monitors and establishes accurate configuration information.

Defining Software Configurations

When a program or module is translated from source to executable, all of its related components are collected from their resident libraries (copylibs, maclibs, and so forth). In order to keep an accurate record of changing software configurations, the process of tracking program-component relationships and associated change activity must be performed automatically as an integral part of the translation procedure.

The Component Monitor

While a program or module is being translated by a processor, ACM automatically captures the program-component relationships as they are resolved from the selected data sets, as the illustration shows:



The Component List

As output from the translation procedure, ACM automatically produces a snapshot or Component List for each monitored program. This Component List is an internal data structure that preserves a physical profile of the configuration. As illustrated next, the Component List identifies all input program components and where they originated, as well as the output components created as the result of the translation procedure. Footprint information, including the version and level of the component, is also noted, if present.

Configuration Type	Name	Footprint Information	Step	Library
Element	PROGRAM X	v1.3		
Processor Record	COMPLINK	v1.8		
Input	COPYRECA	v2.1	Compile	COPYLIB1
Input	COPYRECB	v2.5	Compile	COPYLIB2
Output	PROGRAM X	v1.3	Compile	OBJLIB
Output	PROGRAM X	v1.3	Link-Edit	LOADLIB

The element information describes the program being generated. The processor information describes the CA Endevor SCM processor used to generate the element. The input components identify the component items referenced and any footprints that exist in the monitored data set. The output components identify the elements that were created during processor execution.

Storing Component Lists

The first time a Component List is created, it is stored in its entirety as the base. Subsequent differences in Component Lists from one program translation to the next are stored as deltas (changes only) and automatically assigned a component level number. By using base/delta technology to store Component Lists, ACM enables you to compare component changes from compile to compile, a capability unavailable through source scanning techniques.

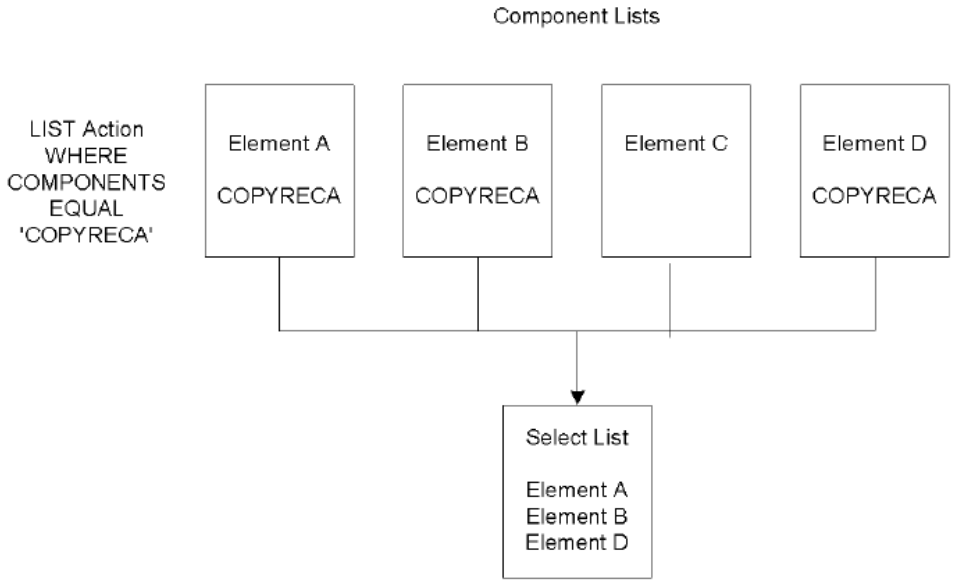
Viewing Component Lists

When stored over time, Component List information becomes the basis for maintaining and viewing component history. Through online displays, ACM lets you view Component List information from four perspectives:

- Summary of component levels
- Current level of a component
- Component changes
- Component history

Using Software Configurations

Using the information contained within Component Lists, it is possible to create a selection list. In the following diagram, a selection list has been created by using CA Endevor SCM to LIST all programs WHERE COMPONENTS EQUAL 'COPYRECA':



As a powerful cross-reference facility, the LIST action, in combination with selection lists in SCL format, provides the necessary foundation for performing a number of essential configuration management functions.

Propagating Component Changes to Related Programs

When a component is changed, it is necessary to propagate those changes to all affected programs. Using the LIST action, you can perform change impact analysis by identifying every program containing that changed component. And you can also use the resulting selection list to perform bulk data manipulation by re-compiling those programs, all at the same time, to reflect the change.

Validating a System for Consistent Use of Components

In order to prevent production failures, it is desirable to validate that all programs are using the correct component versions/levels. Execution reports from LIST action execution, in combination with selection lists, can be used to identify inconsistencies.

Re-Creating Past Configurations

It is sometimes necessary to recreate a program exactly as it has existed at a past point in time. Using the LIST action, you can recreate an older version/level of a program's load module that includes the old versions/levels of all related components used at the date/time the load module was originally created.

Component Validation

Component validation provides the assurance, prior to a promotion, that all dependencies such as Copy Member, Include Member, macros, and so on are included in the promotion and that all programs have been compiled with the current iteration of the copy members.

Release Management

Release management activities revolve around identifying and distributing software releases. Depending on the circumstances, a "release" can consist of the elements changed during a maintenance effort, or it can represent an entire cross-section of the software inventory. Distribution of a release can be as simple as promoting a change into production, or as complicated as managing multiple concurrent releases of entire software systems.

Accurate and up-to-date information is a prerequisite for performing every task, from creating a list of exactly what is to be released to verifying that the release was successfully accomplished. This kind of information can only be obtained through an automated, integrated release management system.

Automated Release Management

In many installations, production turnover and release management are performed, often manually, as discrete activities by separate groups. Problems arise when software release packages are created manually "on paper," where there is a chance that the contents are incomplete or inadvertently changed prior to or during execution.

CA Endevor SCM provides integrated, automated release management functionality, ensuring that software release packages are built on a solid inventory management structure, created using automated techniques, and implemented as an integral part of the release process.

In addition, CA Endevor SCM's change administration facility provides online release package approval capabilities. Only with these combined capabilities is it possible to accurately create and manipulate a software release, perform change impact analyses, and track changes from release to release.

Change Administration

CA Endevor SCM provides change administration, in addition to security, as a means of specifically organizing and controlling change to portions of the software inventory. Through the CA Endevor SCM change administration facilities, it is possible to provide authorization capabilities to the person(s) responsible for portions of the software inventory.

Approver Groups

CA Endevor SCM establishes change administration authorization through the definition of approver groups (named groups of user IDs) and the subsequent relation of those approver groups to portions of the CA Endevor SCM inventory. For example, within the Payroll subsystem of the Finance system, you could establish an approver group that approves or denies changes to all elements within that system and subsystem.

Some of the users within the group could be required to approve changes while the approval of other users would be optional. A quorum or minimum required number of approvers can be established for any approver group. "EMERGENCY" approver groups can be established consisting of those user IDs which must authorize emergency fixes to a portion of the inventory.

Change Packages

Authorized users build change packages, which contain CA Endevor SCM action requests. The user specifies whether the handling of that change package will be defined as 'STANDARD' or 'EMERGENCY'. The user also defines a date/time window within which the change package can be executed.

Once the change package is built, it is then CAST, a process which causes CA Endevor SCM to prepare the package for subsequent approval processing. CA Endevor SCM automatically builds the list of approvers for a change package based on the previously defined approver groups. The change package then goes through approval processing, followed eventually by execution.

How Release Management Works

Release Management works as follows:

Building a Change Package

Authorized users build change packages consisting of any number of CA Endevor SCM action requests. Change packages can be built from online screens, imported from outside data sets and/or copied from existing packages. They can be edited, modified and/or appended to at any point during the build process.

Preparing a Change Package for Review

The change package is CAST. The process of casting a change package causes the approvers for that package to be built based on the previously defined approver groups. Once a package is CAST, it can no longer be modified.

Reviewing a Change Package

Users in the approver group(s) associated with a package can approve or deny the change package. Required users must provide their approval. Once all required approvals have been made, and the established quorum of approvals has been met, the change package can be executed.

Executing a Change Package

At execution time, CA Endevor SCM validates that all approvals have been made and that none of the elements have changed between the time the change package was CAST and when it was executed. CA Endevor SCM also validates that the change package is being executed within the date/time window established when the change package was built. CA Endevor SCM then executes all of the actions in the change package. Checkpoint/restart facilities are available so that change packages that fail execution can be re-executed properly. Change packages can be executed online or in batch.

Automatic Package Backout

Subsequent to execution, it is possible to quickly and easily back out a change package. BACKOUT has the effect of reversing any outputs created by the execution of that package to the state they were in prior to package execution. BACKIN reverses the process.

Managing Production Turnover

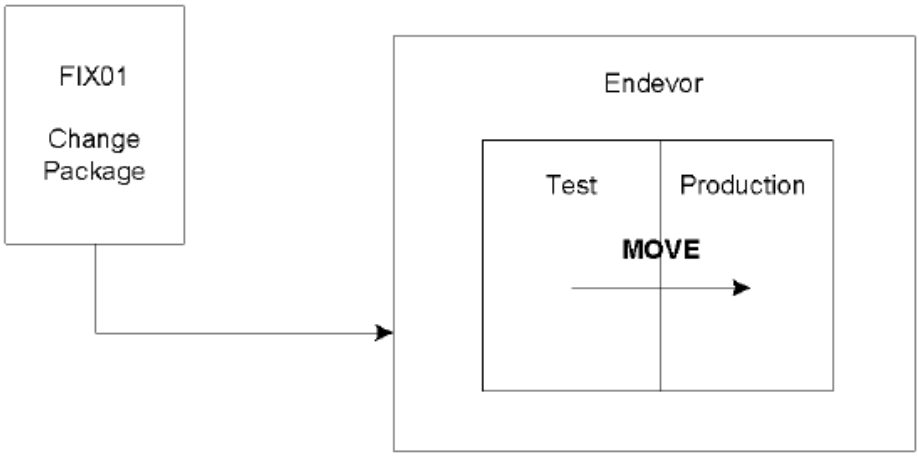
Using the Component Lists created by CA Endevor SCM's Automated Configuration Manager in combination with CA Endevor SCM's logical inventory structure, you can build a change package containing all of the programs and components that make up a maintenance release. Once the package has been built, it can be used to drive production turnover for that maintenance release.

Creating a Maintenance Change Package

When Change Control Identifiers (CCIDs) are used to tag all of the changes which make up a maintenance release, those CCIDs can be used as selection criteria to produce a package that itemizes all of the elements which comprise that maintenance release, as well as all related element components. For example, you could build a package that contains MOVE commands for all of the elements changed for CCID 'FIX01' as well as all the input components for those elements.

Promoting a Maintenance Release

Once the change package has been constructed and has gone through the appropriate approvals, it can be used to perform the production turnover. In our example, execution of the ' FIX01' change package will automatically perform production turnover, promoting all of the elements contained in the package, as illustrated in the following diagram:



Managing Software Releases

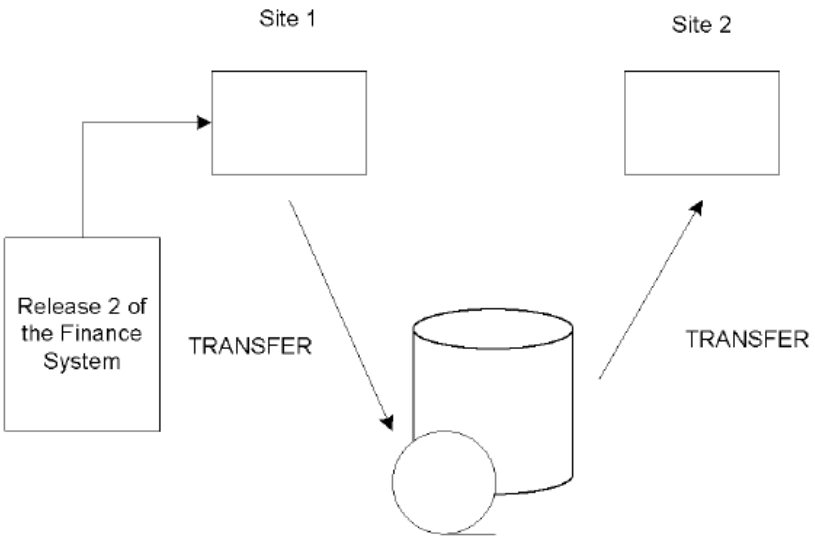
CA Endevor SCM's powerful inventory structure, together with ACM's Component Lists, can be used to manage all aspects of releasing software. Once a change package has been generated, its contents can be used to drive the distribution of the software release, track changes from release to release, and recreate previous releases.

Creating a Software Release Package

The inventory management structure of CA Endevor SCM and the configuration information created by ACM can be used to create complete, reliable, and reusable software release packages. For example, a complete release package could be built containing all of the elements in Release 2 of the Finance system, and all of the components belonging to those elements. The resulting package would contain not only all of the elements that comprise the software release at this point in time, but information about the current versions and levels of those elements.

Release Distribution

Once a release package has been constructed, it can be used to drive the distribution process, regardless of whether the software is to be moved to another CA Endevor SCM location in-house, transferred to another CA Endevor SCM location at a remote machine or site, or distributed to external files, perhaps for shipping to customers or divisions. In our example, the contents of the package created previously are used to drive the TRANSFER of Release 2 of the Finance system from Site 1 to Site 2, as illustrated in the following diagram:



Using the contents of release packages to drive the distribution process provides the added advantage of automating both the distribution process and the documentation of that process.

Release to Release Comparison

An important aspect of any type of release management system is its ability to track changes from release to release. This broad, system-wide perspective is necessary when managing multiple software releases. Packages are kept in such a way that once created, their contents can be used in a number of ways. They can be compared to find out exactly what has changed from one release to the next. If the contents of the package are extracted and stored in CA Endevor SCM's base/delta format, it is also possible to view changes to a software system over time. This capability is especially useful when trying to identify which modules of a software system have been added, removed, or changed prior to distributing a partial release consisting of changed components only.

Supporting Functionality

In addition to its inventory management, change control, configuration management, and release management facilities, CA Endevor SCM provides capabilities in the areas of parallel development management, software security management, and software information management. This broad range of product functionality is designed to support the entire software management lifecycle, not just one aspect.

The Parallel Development Option

CA Endevor SCM provides parallel development management capabilities through the CA Endevor SCM Parallel Development Option (PDM). Using this powerful development tool, it is possible to create a merged program from the comparison of a base program and two independently modified versions of that base program. By eliminating many of the time-consuming, manual tasks associated with merging parallel development updates, PDM saves time and reduces the margin for error.

PDM also provides a means of identifying and resolving conflicts before merging program updates. For this reason, PDM is a necessary tool for effectively managing concurrent application development and resolving problems which arise when integrating in-house modifications with vendor updates.

Managing Parallel Development Activities

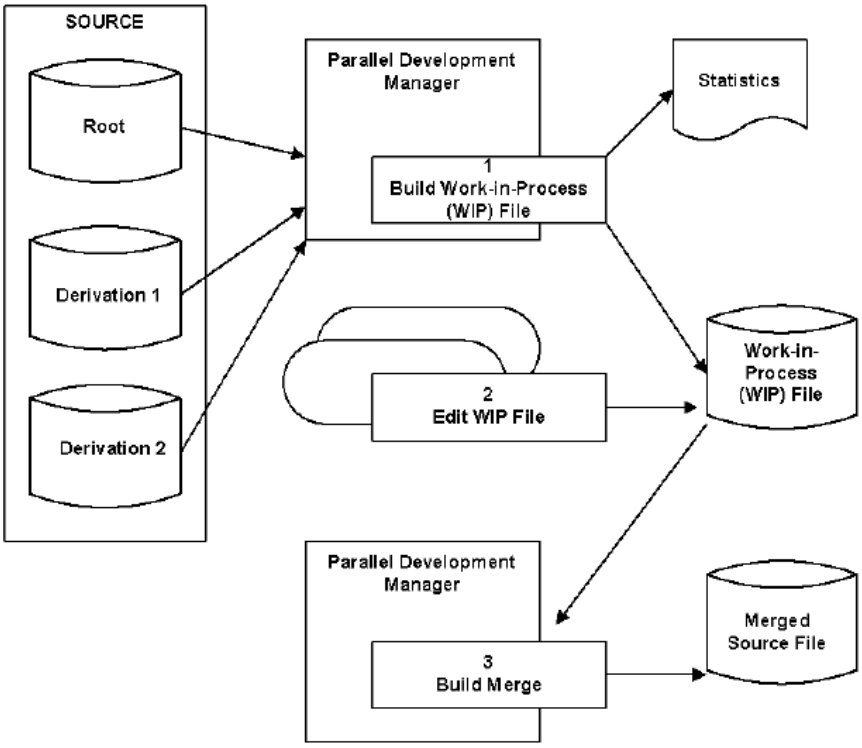
When several programmers work on the same module concurrently, there is always the danger that some modifications will be overridden by others or will be in direct conflict with one another. Beyond the basic frustration factor, this problem can have serious consequences from a control and maintenance standpoint.

Managing Vendor Updates

Purchased software packages are typically customized to meet in-house demands. Unless those modifications are painstakingly documented, problems arise when an update arrives from the vendor. What exactly was changed in-house? And what modules did the vendor change, add, or delete? Manual methods for resolving these conflicts are time-consuming and error-prone, underlining the need for automated solutions for merging vendor updates with in-house modifications, and reconciling changes from release to release.

Basic PDM Operation

The following diagram provides a simple illustration of how PDM operates:



In Step 1, the Work-in-Process (WIP) file is created and statistics are reported. In Step 2, the WIP file is edited. In Step 3, the merged member is built.

Building the WIP File

The first step in using the Parallel Development Manager is to build a Work-in-Process (WIP) file. The WIP file is a PDS structure which combines three source files: the Root, Derivation 1, and Derivation 2. As an example, if PDM is being used to manage a vendor update, the Root would be the source for the old release, Derivation 1 would be the source for the in-house modifications made to that old release, and Derivation 2 would be the source for the vendor's new release.

As the source input files are being compared to produce a WIP File, statistics are generated. These statistics reveal critical information about the source comparison, such as the number of inserts and deletes per module, the percentage of the module that has changed, and the number of simultaneous (and potentially conflicting) updates.

Editing the WIP File

With statistics in hand, it is now possible to view and edit the members which contain conflicts. At this point, you can decide which conflicts to resolve and how to resolve them.

Creating Merged Source

Once you have viewed and edited the WIP file to your satisfaction, you can use PDM to automatically merge the WIP source into an output source library. This output source is then ready for compilation or, if applicable, introduction into CA Endevor SCM. The Parallel Development Manager can be used in both online and batch modes, and can operate on a member-by-member basis on a list of members, or on any portion of the CA Endevor SCM inventory.

Software Security Management

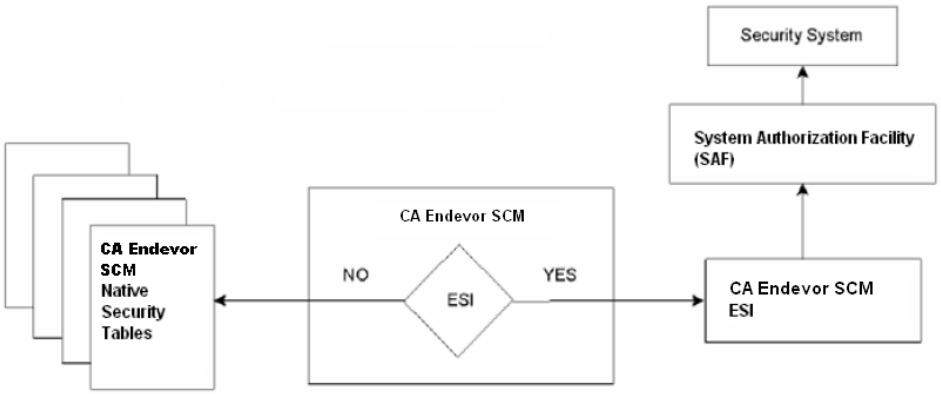
Traditionally, inventory security has been provided at the physical file level but not at the individual element level. CA Endevor SCM security is based on its inventory structure, enabling security rules to be defined according to that structure, at the system, subsystem, type, and element levels. Additionally, CA Endevor SCM actions are secured, providing a means of restricting individuals to the appropriate change control functions at specific stages within the software development lifecycle.

Native Security

CA Endevor SCM native security provides protection against unauthorized access and processing through three tables: Access Security, User Security, and Resource Security. The Access Security Table defines the environments to which each user has access. The User Security Table defines the systems/subsystems within a particular environment and the level of activity for which each user is permitted access (display only, delete, add, and so on). The Resource Security Table defines any element naming conventions that are restricted to a particular system/subsystem and the CA Endevor SCM actions for which the restriction applies. The use of CA Endevor SCM native security is optional and is enabled only if one or more of these tables are defined. When defined, access is permitted only through these tables.

The Interface for External Security

In addition to native security, CA Endevor SCM provides the CA Endevor SCM Interface for External Security (CA Endevor SCM ESI). With CA Endevor SCM ESI, installations which currently use CA ACF2, CA Top Secret, or RACF security systems can secure access to CA Endevor SCM functions and data sets through those centralized systems rather than locally through CA Endevor SCM security tables. CA Endevor SCM ESI accomplishes this through an interface to IBM's System Authorization Facility (SAF). As illustrated in the following diagram, CA Endevor SCM ESI replaces standard CA Endevor SCM security tables with an SAF interface module:



With CA Endevor SCM ESI in place, each security request is checked through the centralized security system in order to validate access to CA Endevor SCM environments, menu options, and actions against inventory elements.

Software Information Management

Through a combination of its master inventory data base (Master Control File) and SMF log file, which notes every change attempted and completed, as well as security violation information, CA Endevor SCM provides extensive display and reporting facilities. The resulting information provides managers with an accurate and reliable means of performing workflow analysis, project tracking, trouble-shooting, and audit management.

Displays

CA Endevor SCM provides extensive display capabilities for viewing status and change information online. The following displays can also be printed in hardcopy format:

- Element Master Control File Information
- Element Change Summary
- Element Change History
- Element Changes
- Element Browse
- Element Output Listing
- Component Summary
- Component History
- Component Changes
- Component Browse
- Footprint Display

Reports

CA Endevor SCM provides four types of reports: Master Control File, Historical, Footprint, and Change Administration.

Master Control File Reports

The following reports reflect the definitions of systems, subsystems, element types, and elements, as specified to the CA Endevor SCM Master Control File:

- System Inventory Profile
- System Inventory Summary
- Element Catalog

- Element Activity Profile
- Element Activity Summary
- Element Catalog by CCID
- System Definition Profile
- Element Signed Out Profile-By System
- Element Signed Out Profile-By User
- Approver Group Definition
- Approver Group Usage

Historical Reports

The following reports summarize security violations and element activity as recorded by CA Endevor SCM:

- Security Violation Profile
- Security Violation Summary
- Element Activity Profile
- Element Activity Summary

Footprint Reports

The following reports document footprint information placed in source and load modules by CA Endevor SCM:

- Library Member Footprint Report
- Library CSECT Listing
- Library ZAPped CSECT Report
- Footprint Exception Report

Change Administration Reports

The following reports document change package activity:

- Package Detail Reports
- Package Summary Report

CA Endevor SCM Benefit Summary

Based on advanced change control technology which is implemented through an automated, integrated approach, CA Endevor SCM yields a number of substantial benefits by doing the following:

- Providing extensive inventory management capabilities which accommodate all source and executable code, regardless of type or record length.
- Providing a means of categorizing, viewing and manipulating physical inventory components as they relate to logical workgroups.
- Creating an unbreakable audit trail of all change events for purposes of historical analysis and auditing.
- Capturing and storing all change information using advanced, space-saving base/delta storage techniques.
- Assuring an inviolate link between executable code and its originating source.
- Enforcing standard, consistent change processes.
- Providing an accurate means of creating and tracking software configurations.
- Automating the movement of entire software release packages throughout the development lifecycle.
- Providing additional approval and notification capabilities through change administration functionality.
- Extending security to the inventory member level.
- Providing a means to compare and implement vendor application updates.

Customization Tables

To help you to customize your implementation, assembler source modules (referred to as tables) are supplied. Your site may not require some of these tables, depending on the options and features selected for your implementation.

Source samples of each of these tables are provided in the installation data *iprfx.iqual.CSIQSRC*. The SMP/E install process will assemble and link-edit all these tables to the user authorized load library (*iprfx.iqual.CSIQAUTU*) that was specified in the installation job BC1JJB01. The Defaults (C1DEFLTS), Email ID Table (ESMTPTBL), Optional Features (ENCOPTBL) and ISPF Configuration (ENDICNFG) tables must be available when CA Endevor SCM is initialized. However, only the Defaults table, C1DEFLTS, must be customized before it can be used. For more information about the Defaults table, see The Defaults Table.

Note: C1DEFLTS, ENCOPTBL, ESMTPTBL, and ENDICNFG are validated at CA Endevor SCM start-up to ensure the tables are release compatible.

The following table describes each table and identifies the corresponding C1DEFLTS parameter in the Defaults table.

Delivered Table Name	Description	C1DEFLTS Parameter
ACCSTABL	Native Security Access Table	ACCSTBL
BC1PNM60	CA Netman Log Program	N/A
BC1TNEQU	ESI Security Definition Table	ACCSTBL
BC1TNM90	CA Netman Start-Up Definition Program	N/A
C1DEFLTS	Defaults Table	N/A
C1GTAPGM	Authorized Program Table	N/A
C1LIBRSQ	CA Librarian Language Definition Table	N/A
C1PTCNTL	CA Panvalet ++CONTOL Password Interface Program	N/A
C1UEXITS	User Exits Table	EXITTBL
ENCOPTBL	Optional Features Table	OPTTBL
ENDICNFG	ISPF Configuration Table	CNFGTBL
ESMTPTBL	Email ID Table	ESMTPTBL

Delivered Table Name	Description	C1DEFLTS Parameter
ESYMBOLS	Site Symbolics Table	SYMBOLTBL
RSCTTABL	Resource Security Table	RSCETBL
USERTABL	User Security Table	USERTBL

Authorized Program Table

The Authorized Program Table (C1GTAPGM) is a table of program names that are eligible to be called by the automated process driver in an authorized mode. If you have non-CA Endevor SCM programs that are invoked within a processor and need to run APF-authorized, then add these programs to the table. Unless you change the table, you do not need to edit the source at all. If changes are required to the table, it must be assembled and link edited using the CSIQJCL member BC1JTABL. It should be included in the STEPLIB of your JCL. The SMP/E install process installs (assembles and links) the C1GTAPGM table into *iprfx.iqual*. CSIQAUTU.

The C1GTAPGM table is mandatory and needs to be define to Endevor whether you modify it or not.

The Defaults Table

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

The Defaults Table should reside in an authorized data set.

For more information, see Using the Defaults Table (see page 207).

Element Actions

You manipulate CA Endevor SCM inventory by executing CA Endevor SCM commands called actions. Some actions are available in both foreground and in batch, while others are available only in batch. Batch actions are also available when you build packages. For more information, see the following guides:

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

The following summarizes CA Endevor SCM actions and their availability:

Add

(Foreground and batch). Puts a member under CA Endevor SCM control from an external data set.

Archive

(Batch) Writes the current version of an element to a sequential data set.

Copy

(Batch) Copies an element from an archive data set to a data set external to CA Endevor SCM.

Delete

(Foreground and batch) Erases base and delta forms of an element and removes related information from a Master Control File.

Display

(Foreground) Displays information about an element.

Generate

(Foreground and batch) Creates an executable form of an element.

List

(Batch) Creates a list of elements that meet specific selection criteria. One effective use of this function is to perform impact analysis.

Move

(Foreground and batch) Moves elements between stages, within or across environments.

Print

(Foreground and batch) Prints element or member information.

Restore

(Batch) Restores elements to CA Endevor SCM from an archive data set.

Retrieve

(Foreground and batch) Copies elements from CA Endevor SCM to an external data set.

Signin

(Foreground and batch) Removes the user signout associated with an element.

Transfer

(Batch) Moves elements between locations that are not on the same map route.

Update

(Foreground and batch) Updates an element from an external data set

Element Actions by Job Function

A typical site might include the following job functions:

- Development
- QA/Test
- Turnover
- Audit
- Management
- CA Endevor SCM administration

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

Action	Dev	QA/Test	Turnover	Audit	Mgmnt	Admin
Add/Update	X	X				X
Archive						X
Copy						X
Delete			X			X
Display	X	X	X	X	X	X
Generate	X					

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

Reporting

CA Endevor SCM provides a full set of standard reports.

Note: For more information about reporting, see the *Reports Guide*.

Source and Output Management

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

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

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

Creating Executable Forms of Elements

Audit Stamps

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

Packages

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

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

Note: For more information about packages, see the *Packages Guide*.

Security

CA Endevor SCM provides two functional security options:

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

A native security facility comes with CA Endevor SCM. It enables you to secure CA Endevor SCM functions (access and actions) by using security tables.

Note: For more information about native security, see the *Security Guide*.

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

Note: For more information about enabling and using CA Endevor SCM ESI, see the *Security Guide*.

CA Endevor SCM and Data Set Security

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

- RACF
- CA ACF2
- CA Top Secret

We recommend that you implement data set security to prevent unauthorized access to the data sets controlled by CA Endevor SCM.

Note: For more information about how to accomplish this using your installation security package, see the *Security Guide*.

Other Capabilities

In conjunction with other CA products, CA Endevor SCM can provide:

- Configuration management, using CA Endevor SCM Automated Configuration (ACM).

  Note: For more information, see the *Automated Configuration Guide*.
- Parallel development controls using CA Endevor SCM Parallel Development (PDM).

  Note: For more information, see the *Parallel Development Guide*.
- Automated coordination of all DB2 processes, using the CA Endevor SCM for DB2 Application Manager.
- Footprint synchronization at remote sites.
- Interfaces to:
  - InfoMan
  - CA Roscoe IE
  - CA Panvalet and CA Librarian
  - CA 7 Workload Automation
  - CA Netman

Chapter 2: Logical and Physical Structure

This section contains the following topics:

CA Endevor SCM Logical Structure

CA Endevor SCM helps to manage the software lifecycle by providing a consistent and flexible logical structure for classifying software inventory. There are six components to this inventory structure: environments, stages, systems, subsystems, types, and elements. Environments, stages, systems, subsystems, and types are set up by the CA Endevor SCM administrator. Users act on elements. The following information defines these terms.

Environment

Functional areas within an organization. For example, there might be separate development and production environments. There is no limit to the number of environments that may be defined.

Stage

The stages in the software lifecycle. Each environment always has one or two stages. Each stage is assigned a unique name and ID, representing their place in the lifecycle. For example, TEST and an ID of 1, or QA and an ID of 2. Stages are referred to in this manual as Stage 1 (the first stage in an environment) and Stage 2 (the second stage in an environment). Stages can be linked together to establish unique promotion routes for program inventory within and between environments. These routes make up the map for a site.

System

The applications at a site. For example, there might be financial and manufacturing applications. A system must be defined to each environment in which it is used.

Subsystem

A specific application within a system. For example, there might be purchase order and accounts payable applications within the financial system. Keep in mind that:

- There must be at least one subsystem per system. A subsystem must be defined to each system in which it is used. For example, to create subsystem PO within system Finance in the environments TEST, QA, and PROD, you must define subsystem PO in each environment.
- A subsystem can have the same name as the system to which you define it.

Type

Categories of source code. For example, you might create the following types:

- COBOL—for COBOL code
- COPYBOOK—for copybooks
- JCL—for JCL streams

You must define a type to each stage in which you want to use it.

Element

Partitioned data set (PDS) members, CA Panvalet members, CA Librarian members, or sequential data sets that have been placed under control of CA Endevor SCM. By default, the element name is the member name. Each element is classified by system, subsystem, and type. Its environment and stage determine its location in the software lifecycle.

The Inventory Structure

The CA Endevor SCM inventory structure allows you to do the following:

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

How to Set Up the Inventory Structure

The CA Endevor SCM administrator builds an inventory structure based on the stages in your site's software lifecycle. There are six steps in setting up an inventory structure:

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

Determine the Lifecycle Stages

Software lifecycles are site-specific. For this example, consider a five-stage lifecycle:

```
DEV       UNITTEST   QA   EMER     PROD
```

Determine the Stages for CA Endevor SCM Control

You can decide to put some or all of the stages in your lifecycle under control of CA Endevor SCM. In this example, assume the last four stages of the lifecycle are under the control of CA Endevor SCM:

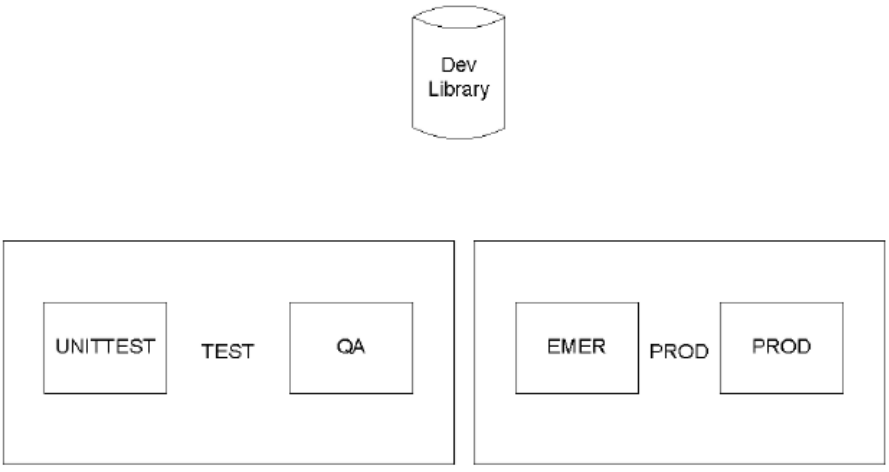
```
UNITTEST   QA   EMER     PROD
```

This means that program development takes place outside of CA Endevor SCM.

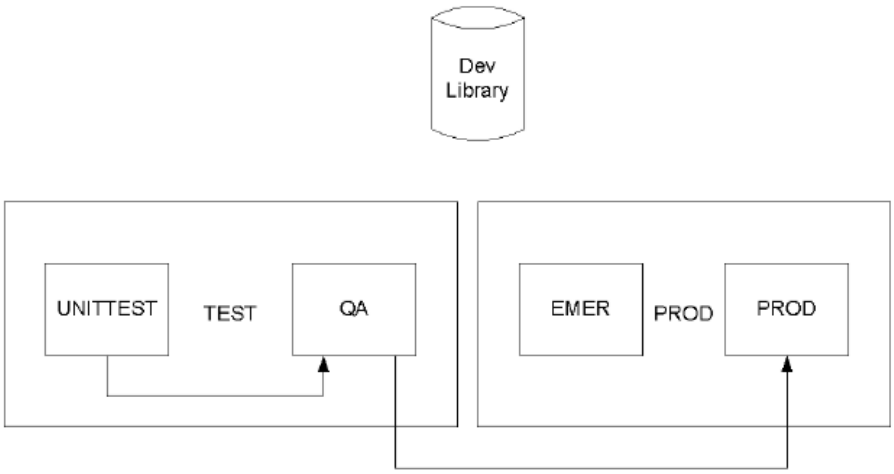
While this is a fairly typical lifecycle, keep in mind that CA Endevor SCM can be adapted to any lifecycle.

Define the Environments

Environment is the CA Endevor SCM term for functional areas in your organization. In this example, assume that the UNITTEST and QA stages in the lifecycle are part of the development function, and that production applications and their maintenance are part of a function called production. The administrator defines environment TEST to include Stages UNITTEST and QA, and a second environment called PROD, that includes Stages EMER and PROD. Development activities take place in a development library, outside of CA Endevor SCM. This is illustrated by the following diagram:



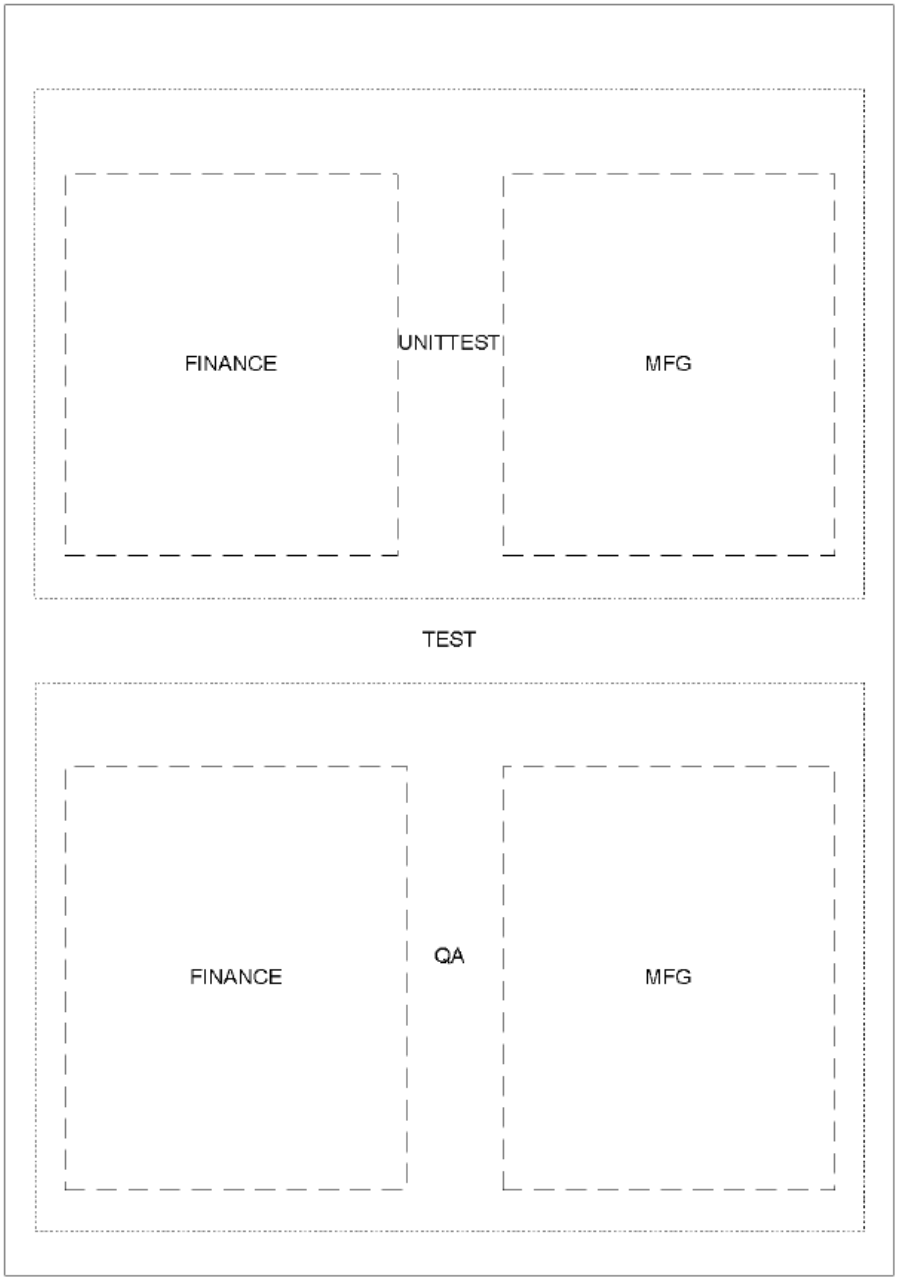
The CA Endevor SCM administrator might decide to establish the following route *(environment map*) for inventory at this site that promotes inventory from Stage UNITTEST to Stage QA to Stage PROD, as illustrated in the following diagram:



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

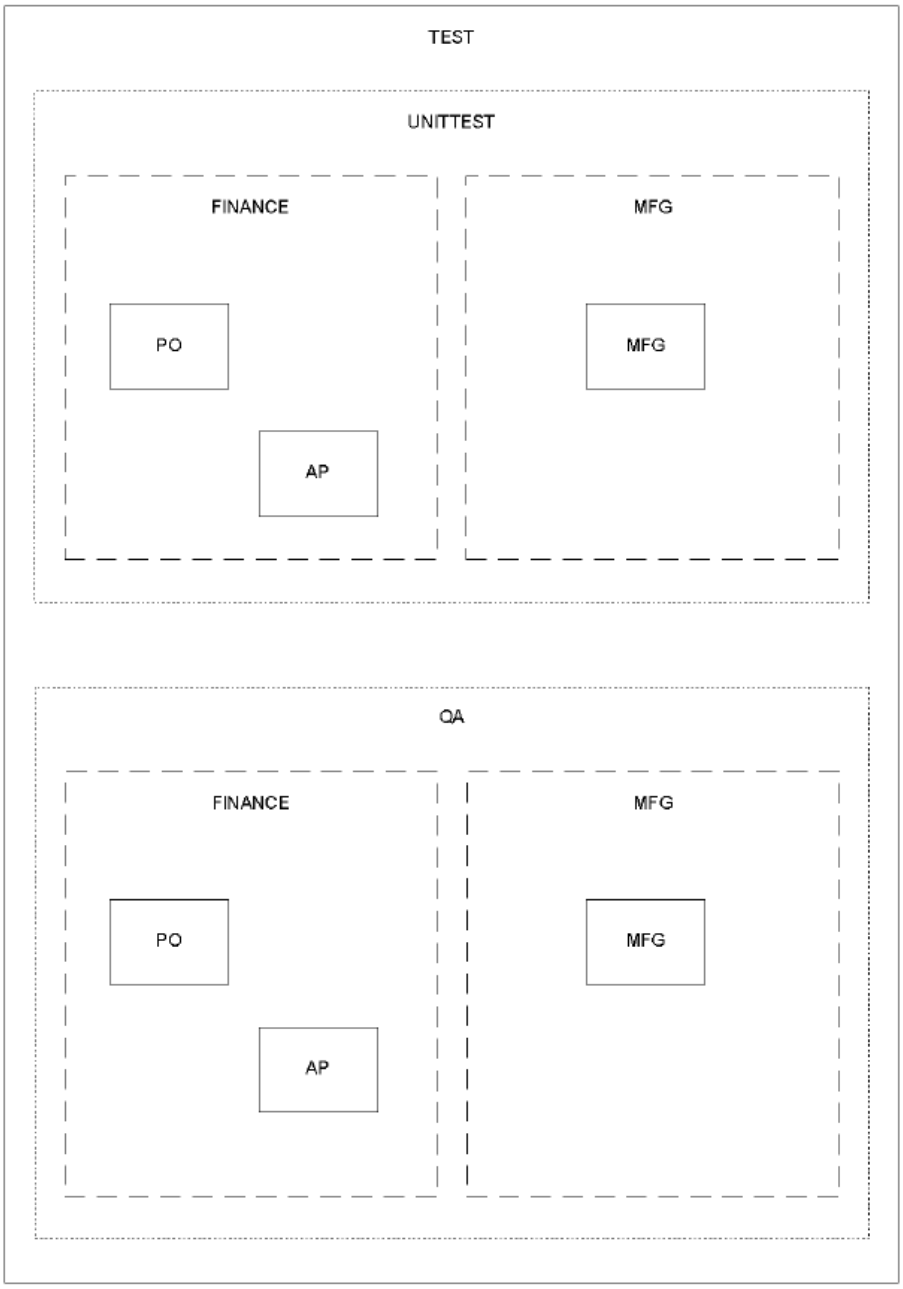
Define the Systems

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



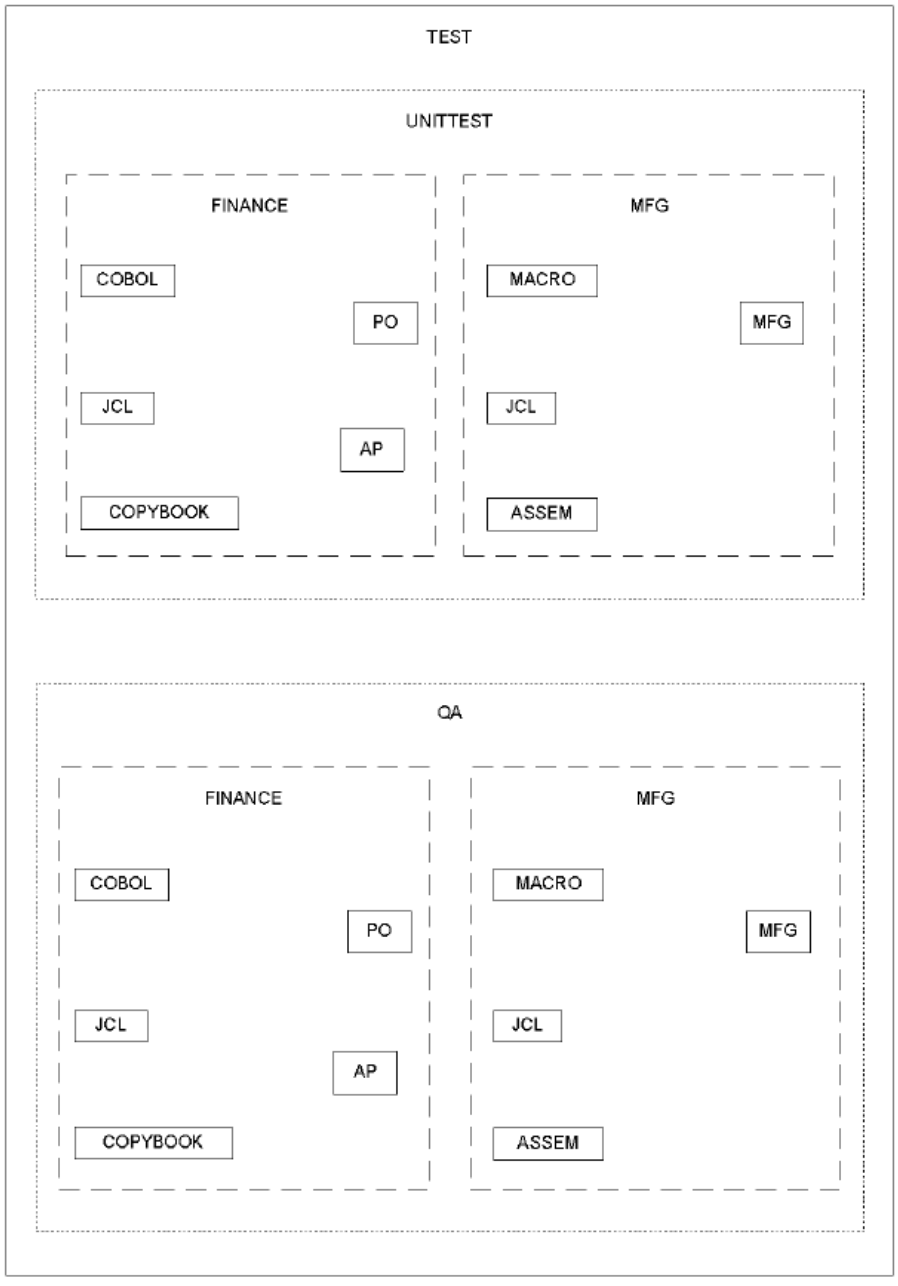
Define the Subsystems

You must define at least one subsystem for each system. In this example, system FINANCE has two subsystems: PO and AP. System MFG one subsystem, MFG. This is illustrated by the following diagram:



Define the Types

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

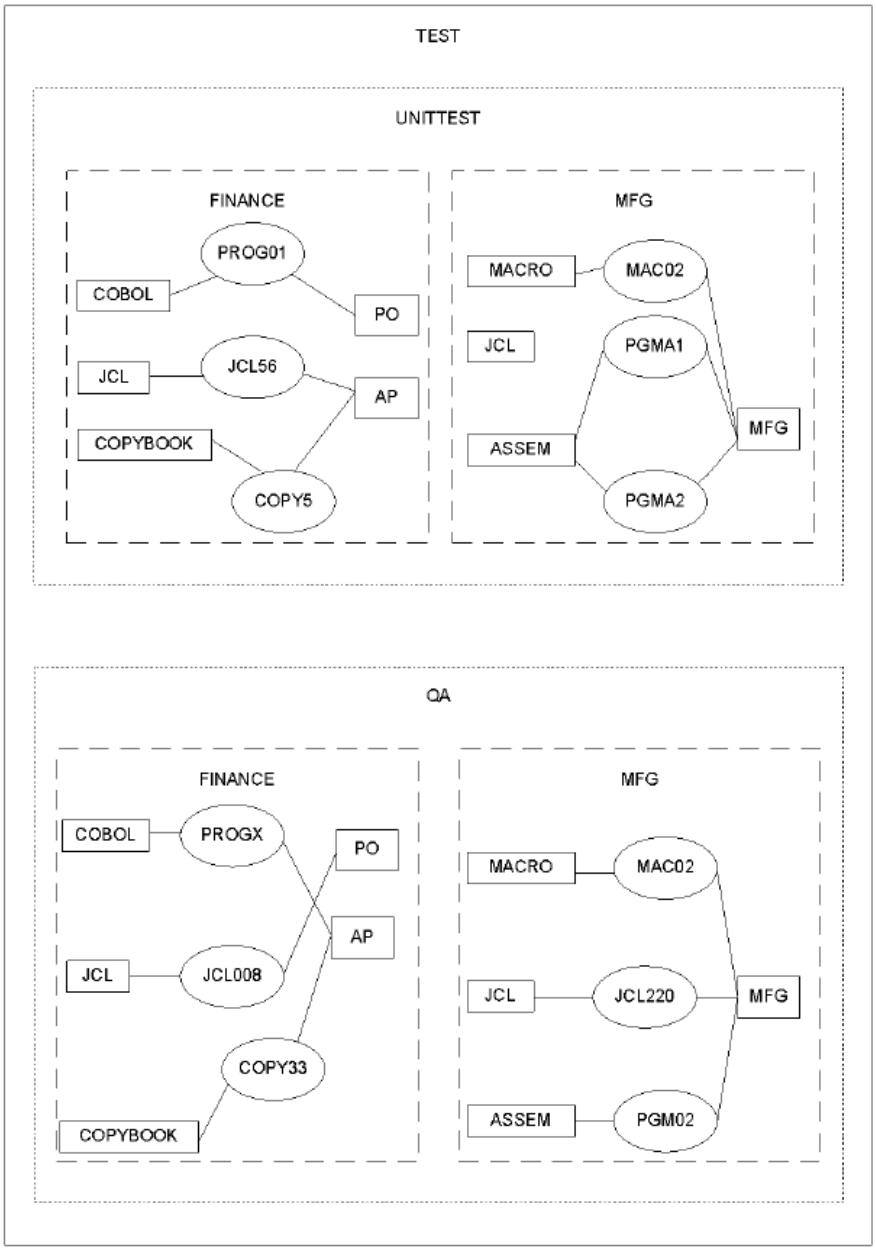


Classify Elements

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

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

This is illustrated by the following diagram:



For example, in the previous diagram, elements are classified as follows:

Module	Location	Classification
PROG01	Environment TEST Stage UNITTEST	Type COBOL Subsystem PO System FINANCE

Module	Location	Classification
JCL220	Environment TEST Stage QA	Type JCL Subsystem MFG System MFG
COPY33	Environment TEST Stage QA	Type COPYBOOK Subsystem AP System FINANCE

Querying the CA Endevor SCM Structure

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

Query	Produces
Show me all the JCL in the shop	JCL56 JCL008 JCL22Q
Show me all the software currently in QA	PROGX JCL008 COPY33 PGM00 JCL22Q MAC02
Show me all the manufacturing software currently being unit tested	PGMA1 PGMA2 MAC02

CA Endevor SCM Libraries

To implement an inventory structure, certain libraries must first be defined and allocated. The following information describes these libraries.

Master Control File libraries

There is one Master Control File (MCF) for every stage. A Master Control File stores system, subsystem, and type definitions, the names of the elements currently in that stage, and other information. Master Control File libraries are defined in the Defaults Table.

Element Catalog

The element catalog is a VSAM KSDS file, similar to the master control files and the package control file. The element catalog enables the support for long or mixed-case element names.

When an element with a long or mixed-case name is added to CA Endevor SCM, the original name is maintained in the element catalog, and a CA Endevor SCM-generated short name (eight characters) is stored in the associated master control file entry. An additional VSAM KSDS file, known as the catalog Cross Reference file, is also maintained, so that actual element names can be retrieved using the short element name that is stored in the MCF.

The element catalog contains one element catalog record for each element-type occurrence and each element catalog record contains a data segment for each location (MCF) where an element resides.

There are limitations associated with the element catalog:

- A given MCF can be associated with only one catalog
- One catalog can be associated with many MCFs
- One catalog can be associated with many C1DEFLTS tables

The key of the element catalog file is 255 characters long. It consists of the following:

1. The first 240 characters of the element name. The remaining 15 characters of the element name are actually stored in the base portion of the record.
2. The element type name

The information supplied to CA Endevor SCM determines whether the element catalog or the MCF file is used during element searches. For example:

- If the system and subsystem names are provided, but *not* the element name—the MCF is searched.
- If an element name is specified—the element catalog is searched; regardless of the system and subsystem specifications.

Package data set

There is one package data set per site. CA Endevor SCM stores all packages built at the site, and related information, in this data set. You define the package data set for your site in the Defaults Table.

Base and Delta libraries

CA Endevor SCM uses base and delta libraries to store source code. Base libraries store source when it is first added to CA Endevor SCM. Delta libraries store changes made to the source. Generally, there is one set of base and delta libraries associated with each type definition. Base and delta libraries are defined during implementation on the type definition panel.

ACM Root and XREF Libraries

CA Endevor SCM uses these libraries to store the name of each element and its related components. This data set is required if your site uses the ACM Query Facility. The ACM library for your site is defined in the C1DEFLTS table.

Note: For more information, see the *Automated Configuration Option Guide*.

Output Libraries

(Processors Only) CA Endevor SCM uses output libraries to store executable forms of programs produced by processors. Allocate these libraries by stage.

You allocate output libraries during CA Endevor SCM implementation.

Source output libraries

CA Endevor SCM uses source output libraries to store copybooks, assembler macros, or JCL procedures that are copied elsewhere and therefore have to be available in full source form.

Note: Source output libraries are type-specific. You can define a source output library for each type in a stage, or share one library across types. Generally, source output libraries are not needed if you store elements in reverse delta format.

You define the source output libraries on the Type Definition panel.

Processor Load and listing libraries

(Processors Only) CA Endevor SCM uses processor load libraries to store the executable form of CA Endevor SCM processors. Allocate one processor load library for both stages of your production environment; point to these libraries from all other stages.

Processor listing libraries are optional. CA Endevor SCM uses them to store listings when processors are compiled.

CA Endevor SCM listing libraries

(Processors Only) Used to store compiler listings produced by the CONLIST utility. A single library can be shared across systems. Listing libraries are *optional* if you do not use CONLIST in your processors. You allocate listing libraries during CA Endevor SCM implementation.

Include libraries

CA Endevor SCM uses include libraries to store the full form of CA Panvalet (++INCLUDE) and CA Librarian (-IN) include statements. You allocate include libraries on the Type Definition panel.

Processor output libraries

(Processors Only) These are libraries that you refer to in processors, to which processors write their output. Processor output libraries can be source libraries, executable libraries, or listing libraries.

Processors and Libraries

The following table summarizes where each of these libraries should be allocated in a sample software lifecycle:

Library Name	C1DEFLTS	DEV TEST	DEV QA	PROD EMERG	PROD PROD
Master control files	X				
Element Catalog	X				
ACM Root library	X				
ACM XREF library	X				
Package data set	X				
Base and delta libraries, by type		X	X	X	X
Output libraries, by type		X	X	X	X
Source output libraries, by type		X	X	X	X
Processor load libraries		X	X	X	X
Processor output libraries (source, executable, list)		X	X	X	X
Include libraries, by type		X	X	X	X

Chapter 3: Defining Inventory Structures

This section contains the following topics:

Basic Panel Flow

This section introduces the basic panels involved in displaying environment and stage information, and defining and maintaining system, subsystem, and type definitions.

Note: For more information about performing these tasks in batch, see the *SCL Reference Guide*.

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

1. Select the environment you want by typing its number in the OPTION field, and pressing Enter.

   CA Endevor SCM displays the Primary Options Panel.

2. Select option **4** (ENVIRONMENT) on the CA Endevor SCM Primary Options Panel, and press Enter.

   CA Endevor SCM displays the Environment Options Menu.

3. To select an option, type the appropriate number or letter (**1-9, A**, **D**, **E**, or **S**) in the Option field, press Enter.

   CA Endevor SCM displays the subfunction panel for the option requested.

4. When you are through, return to the CA Endevor SCM Primary Options Panel by pressing END repeatedly. To return to the invoking facility (ISPF or TSO) from the CA Endevor SCM Primary Options Panel, press End, or type **END** in the Option field, and press Enter.

Note: If the Global Type Sequencing option is enabled at your site, the description on the Environment Options Menu for option 7 TYPE SEQUENCE appears differently than if it is not in effect. When Global Type Sequencing is not in effect, type sequencing can be viewed and updated through the Environment Options Menu. When Global Type Sequencing is in effect, the Type Sequencing can be viewed, but not updated through the Environment Options Menu. For more information, see the *User Guide*.

The Environment Options Menu

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

1

Display site definitions, as specified in the CA Endevor SCM Defaults Table.

2

Display the two stage definitions for an environment.

3

Display, delete, create, update, or clone (copy) the definition and/or mapping of a system.

4

Display, delete, create, or update the definition and/or mapping of a subsystem.

5

Display, delete, create, or update the definition and/or mapping of a type.

6

Display, delete, create, or update the definition and/or mapping of a processor group.

7

Display or update the relative sequence of execution for CA Endevor SCM actions, for all element types defined to a particular system. If the optional feature Global Type Sequencing is enabled at your site, you cannot update the type processing sequence through the Environment Options Menu.

8

Change the data sets defined for use with a specific system.

9

Display, delete, create, or update approver groups.

A

Establish relationships between approver groups and inventory areas within an environment.

D

Display, delete, create, or update destinations for package shipments.

E

Display environment definitions, as specified in the CA Endevor SCM Defaults Table.

S

Display the site symbolics table. The table is specified in the C1DEFLTS table.

Request and Selection Panels

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

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

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

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

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

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

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

Displaying Site Information

This section describes how to display site information.

The Site Information Panel

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

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

Site Information Panel Fields

This section describes the panel fields.

The Customer Name Field

Your company name appears at the top of this screen.

Function Controls Fields

The following information describes the Function Controls fields:

Site ID

ID assigned to the current site.

Release

Volume serial number of the installation tape.

Environments

Number of environments defined at the current site.

Userid Start

First position within a user ID that is compared for ownership (that is, for signout and override signout processing). This field is used when the USERID BATCHID field is **0** (JOBNAME).

Userid Length

Length of the user ID that is compared for ownership. This field is used with the USERID START field when the USERID BATCHID field is **0** (JOBNAME).

Batch ID

Where the user ID associated with a batch job is extracted:

0

From the JOBNAME; the user ID is checked for validity using the userid start and length parameters as established previously.

1

From the USER parameter specified on the job card submitted with the job. If no USER parameter is defined, you receive an error message.

2

From the USER parameter if it is specified on the jobcard or, if no USER parameter has been specified, from the JOBNAME. The user ID is validated using the user ID start and length parameters as established above.

SPFEDIT QNAME

Queue name used when CA Endevor SCM issues an enqueue on a sequential or partitioned data set (not RECFM=U), to prevent simultaneous update to that data set. The data set may be a source library, object library, or other user library. The resource name for the enqueue is the data set name.

SYSIEWL QNAME

Queue name used when CA Endevor SCM issues an enqueue on a PDS defined with RECFM=U (for example, a load library), to prevent simultaneous update to that PDS. The resource name for the enqueue is the data set name.

Authorized Tables

Indicates whether CA Endevor SCM's security tables have to be loaded from authorized libraries. Values are:

Require

Authorized libraries are required.

Allow

Unauthorized libraries are allowed, but CA Endevor SCM issues a warning message.

Ignore

CA Endevor SCM does not check the library's authorization.

Alternate ID

Indicates the value specified for the CA Endevor SCM alternate user ID.

For more information, see Data Set Security in the chapter "Implementing Data Set Security" in the *Security Guide*.

PITR Journal Grp

An L-Serv journal group ID that relates the package data set named in this macro to a specific set of L-Serv journal files. Specification of this parameter enables journaling of changes to Master Control Files and the Package Control File. The format is: (gggg,nnnn), where:

gggg

The journal group ID associated with the package journal files.

nnnn

The journal group subsystem ID.

Note: For more information about the journal group subsystem ID, see the *Utilities Guide*.

CA L-Serv JRNL SBS

The subsystem name associated with the L-Serv address space. This field must be specified if L-Serv is used to control one or more CA Endevor SCM control files and the default L-Serv subsystem name is not used.

Symbolics Table

Name of a load module containing the site symbol definition table created by the user.

User Exits Table

Name of the user exits table.

Options Table

Name of the options table.

Config Table

Name of the configuration table.

SMTP Table

Name of the SMTP table.

Access Table

Name of the Access Security Table currently in use (applicable for native security).

SMF Record Number

Record number assigned to SMF records written out by CA Endevor SCM.

Library System

Indicates the library management system at your site:

LB

CA Librarian and PDS

PV

CA Panvalet and PDS

Library Program

Applicable only if your library management system is CA Librarian (LB). This entry indicates the name of the CA Librarian load module for your site.

VIO Unit

Symbolic device name for temporary disk data sets that are stored on a virtual I/O unit.

Work Unit

Symbolic device name for temporary disk data sets that are not stored on a virtual I/O unit.

Work Volser

Volume serial number of the disk used to store temporary data sets.

Lines per page

Number of lines printed per page, for reports generated by CA Endevor SCM.

MODHLI

Indicates the prefix ,other than SYSyyddd, that is assigned to a temporary data set, creating a pseudo-temporary data set.

This applies to those temporary data sets that are allocated with DISP=MOD in any processor step. Regular temporary data sets, which are not DISP=MOD, use the standard z/OS temporary data set name. This prefix appears as the first node of the data set name. In addition, this parameter affects security for client programs that access the API through Web Services, such as the Eclipse-Based UI, CA CMEW, or user-written client programs.

MODHLI is set in the C1DEFLTS table. For more information about MODHLI, see The TYPE=MAIN Macro (see page 208).

Signout on fetch

Indicates if the fetched element is signed out to you. Valid values are:

Y

The element is signed out when it is fetched, unless it is currently signed out by another user.

N

The element is not signed out.

This value affects Add (Fetch), Generate (Copyback), Move (Fetch), Transfer (Fetch), Search and Replace (Fetch) and Quick-Edit.

Mixed Format

Indicates whether CA Endevor SCM accepts mixed-case entries in CCID, COMMENT, and DESCRIPTION fields. Values are:

CCID

Accept mixed-case in CCID fields.

Comment

Accept mixed-case in COMMENT fields.

Description

Accept mixed-case in DESCRIPTION fields.

All

Accept mixed-case in all three fields.

None

Do not accept mixed-case in any field.

Listing ID string

Character string used against an element's output data sets to identify its listing data set.

Source Sync Check

Sync Check option.

Sync Sev Msg

Sync check error message severity.

Options Fields

The information coded in this section indicates whether you have additional CA Endevor SCM facilities (such as ACM or ESI) in use at your site at this time. The following information describes the Options fields. These fields are display-only fields.

ACM

Indicates if the CA Endevor SCM ACM facility is installed: **Y** or **N**

DB2

Indicates if the CA Endevor SCM for DB2 facility is installed: **Y** or **N**

QUICKEDIT

Indicates if the CA Endevor SCM Quick Edit facility is installed at your site: **Y** or **N**

ESI

Indicates if the CA Endevor SCM ESI facility is installed: **Y** or **N**

INFO

Indicates if the CA Endevor SCM InfoMan Interface facility is installed: **Y** or **N**

LIBENV

This parameter serves two purposes depending upon the LIBENVP parameter in C1DEFLTS.

- If LIBENVP=Y, then
  - LIBENV=PV indicates that CA Panvalet is installed;
  - LIBENV=LB indicates that CA Librarian is installed.
- If LIBENVP=N (default), then the include member syntax is used by CONWRITE, by the Expand Include Utility, or by both as follows:
  - LIBENV=PV indicates that checks will be made for CA Panvalet include member syntax (++INCLUDE) or COBOL COPY statements,
  - LIBENV=LB indicates that checks will be made for CA Librarian include member syntax (-inc) or COBOL COPY statements
  - LIBENV=blank (default), *no* syntax checking or expanding will occur.

NETMAN

Indicates if the CA Endevor SCM Netman Interface facility is installed: **Y** or **N**

PDM

Indicates if the CA Endevor SCM Parallel Development Manager (PDM) facility is installed: **Y** or **N**

PROC

Indicates if you have the ability to run processors at your site: Y or N

Package Processing Fields

The following describes the Package Processing fields. These fields are display-only fields.

Approval Required

Indicates whether packages must be approved: **Y** or **N**

Foreground Execution

Indicates whether packages may be executed in foreground.

High-Level Index for Generated Remote PKG Ship JCL

Controls the choice of the remote JCL generation for package ship.

For more information, see the definition of the parameter RJCLRoot in TYPE=MAIN Macro in the chapter "Using the Defaults Table."

Cast Security

Indicates whether to check security authorizations for every action in a package for the user ID requesting package cast: **Y** or **N**

Inspect Security

Indicates whether to check security authorizations for every action in a package for the user ID requesting package inspect: **Y** or **N**

Comp Validation

Indicates whether component validation is enabled for packages. Values are:

Y

Validation is required.

O

Validation is optional.

N

Validation not required.

Security

Indicates the type of security controlling package actions after CAST.

APPROVER

The site is restricting package actions to package approvers.

ESI

The site is controlling package actions through an external security package such as CA ACF2, CA Top Secret, or IBM RACF using the ESI interface.

MIGRATE

The site is in transition between Approver security and ESI security. Both are checked.

Important! If the user is granted access to the package by the approver rules, ESI is not invoked. ESI is invoked only when the user does not belong to any approver groups associated with the package. If there are no approver groups associated with the package, no access restrictions apply.

Control Data Set Names Fields

These fields are display-only fields and are described in the following:

Element Catalog

Name of the file containing the element catalog.

EINDEX

Name of the file containing the element catalog EINDEX.

Package Control File

Identifies the data set used in this environment to store packages.

CA Endevor SCM Macro Library

Data set name of the source library established for this site during installation. This is the library that contains the CA Endevor SCM macros.

CCID Validation Data Set

Data set name of the sequential file containing the definitions of the valid CCIDs established for this site. This field is blank if CCID validation is not in use.

ACM Query Root Data Set

The name of the VSAM file your site uses to store the name of each CA Endevor SCM element and all its related components. The recommended name is uprfx.uqual.ACMROOT.

ACM Query Xref Data Set

The name of the VSAM file your site uses to store the name of each CA Endevor SCM component relationship. The recommended name is uprfx.uqual.ACMXREF.

CA Endevor SCM Parmlib Information

The following information describes the CA Endevor SCM Parmlib Information fields. These fields are display-only fields.

CA Endevor SCM Parmlib Data Set

Data set name of the library, which contains the Type Sequence Member.

Type Sequence Mbr

The name of the file that defines the processing order for Global Type Sequencing.

CA 7 Workload Automation Data Set Name Fields

The following information describes the CA 7 Workload Automation Data Set Name fields. These fields are display-only fields.

CA 7 Region CCI Nodename

CCI Nodename assigned to the CA 7 Workload Automation address space.

JCL Data Set Index Number

Sequence number associated with the CA 7 Workload Automation DEMAND JCL library.

JCL Data Set Index Symbol

Symbol associated with the CA 7 Workload Automation DEMAND JCL library.

JCL Data Set Name

Data Set name of the CA 7 Workload Automation JCL DEMAND library.

Concurrent Action Processing

The following information describes the fields in the Concurrent Action Processing Values panel. These fields are display-only fields.

Spawn Indicator

Indicates if the Concurrent Action Processing feature is enabled. Allowable values are Y or N.

N (default)

Indicates that concurrent action processing is disabled. Actions will be processed one at a time regardless of any other C1DEFLTS parameters or the presence of the EN$CAP*nn* JCL DD card.

Y

Indicates that concurrent action processing is allowable in applicable circumstances. However, if SPAWNCNT=0 was specified in C1DEFLTS then concurrent action processing will not take place unless an EN$CAP*nn* DD card is present in the JCL.

Spawn Count

The default number of action request regions to be used. To override this value the user can include an EN$CAP*nn* DD card where *nn* specifies the number of processors to be used. The default is 0.

Spawn Count Max

The maximum number of action request regions allowed for a job. If EN$CAP*nn* exceeds this value, the SPAWNMAX value overrides the nn value in EN$CAP*nn*. The default is 99.

Spawn Proc Name

The one to eight character name of the Concurrent Action Processing server. The default value is ENDEVOR.

Displaying Stage Information

This section describes how to display the stage information.

The Stage Information Panel

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

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

Stage Information Panel Fields

The following fields appear on the Stage Information panel:

Current Env

Name of the environment for which the stage definitions are shown. Fill in a new name, and press Enter to display the stage definitions for another environment.

Next Env

Name of the next environment in the map.

Stage ID

ID of the first mapped stage in the next environment.

Stage 1 Information

Note: All Stage 1 information is display-only.

Stage 1 definition information includes the following:

ID

Stage 1 ID

Name

Stage 1 name

Title

Stage 1 title

MCF data set name

Data set name of the Stage 1 Master Control File (MCF).

Entry Stage

Indicates whether Stage 1 is the entry stage.

Stop at Stage

Indicates whether Stage1 is a Stop at Stage for promotion packages.

Stage 2 Information

Note: All Stage 2 information is display-only.

Stage 2 definition information includes the following:

ID

Stage 2 ID

Name

Stage 2 name

Title

Stage 2 title

MCF data set name

Data set name of the Stage 2 Master Control File (MCF).

Entry Stage

Indicates whether Stage 2 is the entry stage.

Stop at Stage

Indicates whether Stage 2 is a Stop at Stage for promotion packages.

Defining Systems

This section describes how to define a new system.

The System Request Panel

To define a new system, select option **3** on the Environment Options Menu, and press Enter. CA Endevor SCM displays the System Request panel.

Create a New System

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

1. Select option **C** or **K** to create a new system.

   Note: For more information about option **K**, see Cloning System, Subsystem, and Type Definitions (see page 96).

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

2. Enter an environment name, if different from the displayed name.

3. Enter a system name or mask.
4. Press Enter.
   - If CA Endevor SCM displays a System Selection List, select a system, press Enter, and proceed to Step 5.
   - If CA Endevor SCM displays a System Definition panel, proceed to Step 5.
5. Type or change information as necessary on the System Definition panel, then press Enter to save the changes.

   Note: A different panel, specific to clone processing, is returned if you request clone processing.

Using the System Definition Panel

The following information illustrates how the use of the System Definition panel varies by processing option:

Display

Display the system definition.

Delete

View the system definition and verify that you want to delete it. Press End if you want to cancel the delete request.

Create

Define a new system.

Update

Change an existing system definition.

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

System Definition Panel Fields

This section describes the panel fields.

Identification Fields

As shown in the following information, the first three fields on the System Definition panel identify the environment:

Current Env

(Display-only) Name of the current environment.

Next Env

(Display-only) Name of the next environment on the map.

System

(Display-only) Name of the current system.

Next System

Name of the system this system maps to in the next environment. You can enter or change the name in this field when you access this panel in create or update mode.

Title

Descriptive title for the system (1-50 characters).

Updated

(Display-only) This field identifies the date, time, and user ID of the last user to update the system. When creating a new system definition this field is blank.

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

General Options Fields

The following information describes the General Options fields:

Comment

Indicates whether there must be a comment for actions against this system. Acceptable values are:

Y

Each action must have a comment.

N

Default. Comments are not required for actions.

CCID

Indicates whether there must be CCIDs for actions against this system. Acceptable values are:

Y

Each action must have a CCID.

N

Default. CCIDs are not required for actions.

Req Elm Jump Ack

Indicates whether users must specify ACKNOWLEDGE ELM JUMP=**Y** on the Move panel when jumping elements. Acceptable values are:

Y

User must specify ACKNOWLEDGE ELM JUMP=**Y**.

N

Default. User does not have to specify ACKNOWLEDGE ELM JUMP=**Y**.

Note: Jumping occurs when you move an element from one stage to another on a map route, and a version of the element exists at an intermediate stage that is not part of the map route.

Auto Age Level Retention Options Fields

The following fields on the System Definition panel identify the auto age level retention options available at the system level. Using these fields, you can activate auto age driven level retention for elements, components, or both.

Element

Indicates whether the element change levels are to be managed by age.

N

Default. Age managed change levels are not in effect.

Y

Age managed change levels are in effect.

Retain Lvls For

Indicates the length of time, in months, that change levels are to be kept. For reverse (image) deltas, expired levels are dropped. For forward deltas, expired levels are incorporated into the base. When age retention is on, allowable values are 1 through 999 and the default is 999 months. Zero must be specified when age level retention is off. Removal of change levels only occurs when an element is moved or added to the end of system map route location.

Component

Indicates whether the component change levels are to be managed by age.

N

Default. Age managed change levels are not in effect.

Y

Age managed change levels are in effect.

Retain Lvls For

Indicates the length of time, in months, that change levels are to be kept. For reverse (image) deltas, expired levels are dropped. For forward deltas, expired levels are incorporated into the base. When age retention is on, allowable values are 1 through 999 and the default is 999 months. Zero must be specified when age level retention is off. Removal of change levels only occurs when an element is moved or added to the end of system map route location.

Note: The auto age level retention options for elements and components can only be switched on or off on the system definition. If these options are activated at the system level, auto consolidation options on the type definitions are ignored for all types under the system.

Element Registration Options

The Element Registration Check Options fields contain definitions of the element registration features in effect for this System. This section of the System Definition panel includes the following fields:

DUP ELEMENT NAME

Specifies whether the duplicate element name check feature is active. Valid values are:

Y

CA Endevor SCM checks to see if the element name is used in any other Subsystems within this System.

N

CA Endevor SCM does not check for the same element name in other Subsystems. The default.

MSG LVL

Specifies the error message severity level if CA Endevor SCM is checking for duplicate element names within the system. Valid values are:

C

If the same element name exists within another subsystem under the same system, the action is performed and a caution message is issued.

W

If the same element name exists within another subsystem under the same system, the action is performed and a warning message is issued.

E

If the same element name exists within another subsystem under the same system, the action is terminated and an error message is issued.

Note: This field must be blank if DUP ELEMENT NAME is set to No.

DUP PROC O/P TYP

Specifies whether CA Endevor SCM checks element names across types and processors groups for the same processor output type. A conflict occurs if you attempt to add or create an element and an element with the same name and processor output Type exists in the same System, with a different Type.

Y

CA Endevor SCM checks if duplicate element names with the same processor output type exist within the same system, but are a different type.

N

CA Endevor SCM does not check if duplicate element names with the same processor output type exist within the same or different systems.

ACROSS SBS=Y

Extends the check for duplicate processor output Types to all the Subsystems defined to this System. The check compares the processor output Type of the current action against all processor output Types of all same named elements in the different Subsystems of the same System. If any are equal, the current action fails with a registration error.

MSG LVL

Specifies the processor output registration check message severity level.

C

An element with the same name and processor output type but different type exists within the same system. The action is performed and a caution message is issued.

W

An element with the same name and processor output type but different type exists within the same system. The action is performed and a warning message is issued.

E

An element with the same name and processor output type but different type exists within the same system. The action is terminated and an error message is issued.

Note: This field must be blank if the Duplicate Proc O/P Type Check feature is set to N.

Signin/Signout Options Fields

These fields contain definitions of the signin/signout functions in effect for the system. The following information describes the Signin/Signout Options fields:

Activate Option

Acceptable values are:

Y

The signin/signout facility is in use for this system.

N

The signin/signout facility is *not* is use.

Note: If signin/signout is set to N, the signout userid field is still updated. If you set this to Y, an ESMPTBL is present, and an element signout is overridden, CA Endevor SCM searches the table for the user ID that the element is signed out to. If a match is found, an email is addressed to the signout user. If no match is found, no email is sent. For more information about the Email Notification facility, see the appendix "Using the Email Notification Facility."

Validate Data Set

Acceptable values are:

Y

The data set validation facility is in use for this system.

N

The data set validation facility is *not* in use.

When an element is retrieved, a "retrieve to" data set name (and member name if the data set is a library) is placed in the element master record.

When an ADD or UPDATE action is performed against an element, CA Endevor SCM compares the "retrieve to" data set name (and member name if applicable) to the source input data set and member name specified for the action.

- If the data set (and member) names match, the action continues.
- If the names do not match, CA Endevor SCM checks the value in the override signout field. If this value is:
  - **Y**-Processing continues.
  - **N**-The action fails.

Last System Backup Fields

These fields contain the date and time of the most recent backup of the system. They appear on the System Definition panel in display and delete modes only. These fields are display-only fields. The following information describes the Last System Backup fields:

Date

Date of most recent system backup.

Time

Time of most recent system backup.

Processor Translation Output Libraries Fields

These fields contain the names of the processor load and list libraries for the system. The following information describes the Processor Translation Output Libraries fields:

Stage 1 Load Library

Name of the Stage 1 processor load library for this system. This load library must be different from the load library for Stage 2.

Stage 1 List Library

Name of the Stage 1 processor listing library for this system. This listing library must be different from the listing library for Stage 2.

Stage 2 Load Library

Name of the Stage 2 processor load library for this system. If this load library is the same as the load library for Stage 1, CA Endevor SCM issues a warning message.

Stage 2 List Library

Name of the Stage 2 processor listing library for this system. If this listing library is the same as the listing library for Stage 1, CA Endevor SCM issues a warning message.

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

Cloning System, Subsystem, and Type Definitions

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

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

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

Clone a New System

Follow this procedure to clone inventory definitions:

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

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

   The System Request panel appears.
5. Repeat Steps 1-4 for each inventory definition that you want to clone.

To Information

The following information describes the To Information fields that appear on the System Definition Panel. These fields identify the environment and system for which inventory definitions are cloned.

Environment

(Display-only) Name of the current environment.

System

(Display-only) Name of the new system.

Title

Description of the new system.

From Information

The following information describes the From Information fields that appear on the System Definition Panel. These fields identify the environment and system from which inventory definitions are cloned.

Environment

Name of the source environment. You can change the environment name to clone a system definition from another environment.

System

Name of the system that is cloned.

General Options

The following information describes the General Options fields that appear on the System Definition Panel. Use these fields to indicate whether or not to clone subsystem, type, and processor group definitions for the named system.

Copy Subsystems

Acceptable values are the following:

Y

Default. Clone subsystem definitions.

N

Do not clone subsystem definitions.

Copy Types

Acceptable values are the following:

Y

Default. Clone type and processor group definitions.

N

Do not clone type and processor group definitions.

Defining Subsystems

This section describes how to create, display, update, and delete a subsystem.

Define a Subsystem

Use this procedure to define a subsystem.

1. Select option **4** on the Environment Options panel, and press Enter.

   The Subsystem Request panel appears.
2. Select an option (**Blank**, **#**, **C**, **U**, or **K**).
3. Enter an environment name, if different from the displayed name.
4. Enter a system name or mask.
5. Press Enter.
   - If CA Endevor SCM displays a System Selection List, select a system, press Enter, and proceed to Step 6.
   - If CA Endevor SCM displays a Subsystem Definition panel, proceed to Step 6.
6. Type or change information as necessary on the Subsystem Definition panel, then press Enter to save the changes.

The Subsystem Definition Panel

Once you have entered the necessary information on the Subsystem Definition panel, press Enter to perform the requested processing. The following information illustrates how the use of the Subsystem Definition panel varies by processing option:

Display

Display the subsystem definition.

Delete

View a subsystem definition and verify that you want to delete it. Press End if you want to cancel a delete request.

Create

Define a new subsystem.

Update

Change an existing subsystem definition.

Subsystem Definition Panel Fields

The following information describes the fields that appear on the Subsystem Definition panel:

Current Env

Name of the current environment.

System

Name of the current system.

Next Env

Name of the next environment in the map.

System

Name of the next system in the map.

System Title

Descriptive title for the system.

Subsystem

Name of the subsystem being processed.

Title

Descriptive title for the subsystem (1-50 characters).

Next Subsystem

Name of the subsystem this subsystem maps to in the next environment and system.

Updated

Identifies the date, time, and user ID of the last user to update the subsystem. When creating a new subsystem definition this field is blank.

Exclude from duplicate element proc o/p Type check

Excludes this Subsystem from the processor output Type check, if the check is active for the System to which this Subsystem is defined.

Defining Types

Types identify categories of code. Types are system and stage specific. You must define types at each stage in an environment. Each system can have 256 types defined to it. For example, if you wish to use type COBOL in both the finance and manufacturing systems, you must define it to both systems in each stage where the systems are defined. If you are using single-stage environments (ENTRYSTG#=2) then you only need to define the type in stage 2.

Define a Type

Use this procedure to define a type.

1. Select option **5** on the Environment Options panel, and press Enter.

   The Type Request panel appears.

2. In the fields on the Type Request panel, type the following and press Enter:

   - An option (**Blank, #, C**, or **U**)**.**
   - An environment name, if different from the displayed name.
   - A system name or mask.
   - A type name or mask.
   - A stage ID.

3. If CA Endevor SCM displays:
   - A System Selection List, select a system, press Enter, and proceed to Step 4.
   - A Type Selection List, select a type, press Enter, and proceed to Step 5.
   - A Type Definition panel, proceed to Step 5.
4. If CA Endevor SCM displays:
   - A Type Selection List, select a type, press Enter, then proceed to Step 5.
   - A Type Definition panel, proceed to Step 5.
5. Type or change information as necessary on the Type Definition panel, then press Enter to save the changes.

A sample Type Definition panel in update mode is shown next:

```
UPDATE  ---------------------- TYPE DEFINITION -------------------------------
COMMAND ===>
CURRENT ENV:  USS2       STAGE ID:  3   SYSTEM:  FINANCE    TYPE:  COB
NEXT ENV   :  USS2       STAGE ID:  4   SYSTEM:  FINANCE    TYPE:  COB

DESCRIPTION   ===> Type for environment USS2 stage 1
UPDATED:           14FEB13 07:51 BY USERID
         ---------------------  ELEMENT OPTIONS  ---------------------
DELTA FORMAT(F/R/I/L) ===> R  SOURCE LEN   ===> 80     ELE RECFM(N/F/V) ===> N
COMPRESS/ENCRYPT(Y/N) ===> N  COMPARE FROM ===> 7      DFLT PROC  ===> COB
AUTO CONSOL (Y/N)     ===> Y  COMPARE TO   ===> 72     LANGUAGE   ===> DATA
CONSOL AT LVL  ===> 96        REGRESSION%  ===> 50     PV/LB LANG ===> COB
LVLS TO CONSOL ===> 50        REG SEV(I/W/C/E)  ===> I DATA FORMAT(T/B) ===> T
HFS RECFM(COMP/CR/CRLF/CRNL/F/LF/NL/V)     ===> NL     FILE EXT   ===>
         -----------------  COMPONENT LIST OPTIONS  ------------------
FWD/REV DELTA(F/R)    ===> F  AUTO CONSOL (Y/N) ===> Y CONSOL AT LVL  ===> 96
                                                       LVLS TO CONSOL ===> 50
         ------------------------  LIBRARIES  ----------------------
  BASE/IMAGE LIBRARY ===> BST.QA.USSSHIP.C11600.STG&C1STGID..BASE
  DELTA LIBRARY      ===> BST.QA.USSSHIP.C11600.STG3.DELTA
  INCLUDE LIBRARY    ===> &#PATHINC
  SOURCE O/P LIBRARY ===>
    EXPAND INCLUDES(Y/N)  ===> Y
```

Using the Type Definition Panel

The following information illustrates how the use of the Type Definition Panel varies by processing option:

Display

Display a type definition.

Delete

View a type definition and verify that you want to delete it. Press End to cancel a delete request.

Create

Define the new type.

Update

Modify a type definition.

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

Type Definition Panel Fields

This section describes the panel fields.

Identification Fields

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

Current Env

Name of the current environment.

Stage ID

Name of the current stage.

System

Name of the current system.

Type

Name of the type.

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

Next Env

Name of the environment at the next map location.

Stage ID

Stage ID of the next stage in the map.

System

Name of the system at the next map location.

Type

Name of the type at the next map location.

Important! Type names cannot be changed across the map.

Description

Displays a 1- to 50-character description of the type.

Updated

Identifies the date, time, and user ID of the last user to update the type definition. When creating a new type definition, this field is blank.

Element Options

The following describes the Element Options fields:

DELTA FORMAT (F/R/I/L)

Specifies delta storage format for elements of this Type. Valid values:

F— Default. Forward delta format.

R— Reverse delta format.

I— Full-image delta format.

L— Log delta format.

Note: For more information about formats, see Element Storage Formats (see page 113).

SOURCE LEN

Logical record length in source statements. The maximum allowable value is 32,000. For variable-length records, this length does not include the four-byte record header.

Note: When updating an existing type, do not modify this value if any elements of that type already exist.

ELE RECFM (N/F/V)

Specifies whether the element has fixed length records or variable length records. This parameter is used by the CA Endevor Quick Edit option only. During a Quick Edit session, a temporary ISPF edit data set is allocated to hold the element to be edited. The edit data set is allocated either as variable or fixed.

If N is specified, the element RECFM is determined by the input data set record lengths. For example, the data set is allocated as variable if the element already exists and the element records have varying lengths. It is allocated as fixed if the element already exists and all the element records are the same length. It is always allocated as fixed if a Quick Edit Create command is executed to create a new element. If it is allocated as fixed length, the record length is the Type source length defined by Source Length parameter. Valid values follow:

F— Fixed. Allocates the temporary Quick Edit ISPF edit data set as fixed.

V— Variable. Allocates the temporary Quick Edit ISPF edit data set as variable.

N— Not defined. Allocates the temporary Quick Edit ISPF edit data set based on input record length. Default.

COMPRESS/ENCRYPT (Y/N)

Indicates whether to encrypt and compress the base form of elements stored in reverse delta format. Acceptable values are the following:

N— Do not compress base and encrypt name.

Y— Default. Compress base and encrypt name.

If you are not using a USS Base library, then the value must be set to Y.

COMPARE FROM

Position within each statement at which CA Endevor SCM begins comparing to identify changed statements (5 digits in the range 1-32,000).

Default: 1.

DFLT PROC

Identifies the processor group for this type. The default is *NOPROC*. When you type or update the name of the processor group, then press Enter, CA Endevor SCM displays a Processor Group Definition panel.

- If the specified processor group exists, you can use the Processor Group Definition panel to verify or modify the processor group.
- If the specified processor group does not exist, you can use the Processor Group Definition panel to create it as a new processor group.

Note: For more information, see the *Extended Processors Guide*.

AUTO CONSOL (Y/N)

Indicates whether CA Endevor SCM is to consolidate change levels automatically in conjunction with Consol at Lvl and Lvls to Consol. Acceptable values are:

Y— Consolidate automatically, when physical Consol at Lvl is reached. For example, if Consol at Lvl is set to 96, and Lvls to Consol is set to 50, when the 96th physical change level is reached, CA Endevor SCM will automatically consolidate the first 50 change levels into a single level, and adjust the level number accordingly.

N— Default. Do not consolidate automatically. The LVLS TO CONSOL field must be set to 0.

Note: If auto age level retention for elements is activated at the system level, the AUTO CONSOL field does not appear. It is replaced by the AUTO RETENT field and the CONSOL AT LVL and LVLS TO CONSOL fields do not appear.

Note: For more information about element change levels (deltas) see Element Storage Formats (see page 113).

COMPARE TO

Position within each statement at which CA Endevor SCM stops comparing to identify changed statements (5 digits in the range 1-32,000). If you plan to use in-stream data in processors, set this value to 80 for type Process.

Default: 72.

LANGUAGE

User-specified. Defines the source language for the type (up to eight characters).

Note: If you specify link-edit in this field, you cannot use name or alias statements in the link step of processors associated with this type.

For the type Process, the language field must be CNTLPROC.

CONSOL AT LVL

Specifies the number of physical change levels at which CA Endevor SCM will perform automatic consolidation.

Default: 96.

For example, if the value in this field is 70, CA Endevor SCM consolidates levels when the number of physical change levels reaches 71.

Note: If age level retention for elements is activated at the system level, the AUTO RETENT field appears in place of the AUTO CONSOL field and the CONSOL AT LVL and LVLS TO CONSOL fields do not appear.

REGRESSION%

Maximum acceptable regression percent for elements of this type (2 digits). The use of a regression percentage of 00 turns off regression testing for that type-no change or base regression messages are issued.

Note: If the delta storage format for this type is I (full-image delta format) or type L (log delta format), the regression percent must be 00.

PV/LB LANG

Required field used for internal purposes as well as with CA Librarian or CA Panvalet. The up to eight character CA Panvalet or CA Librarian source language for the type. This field must be in agreement with the LIBENV specification defined in the C1DEFLTS table. For more information about LIBENV, see The TYPE=MAIN Macro (see page 208).

Acceptable values for CA Panvalet include the following:

- ANSCOBOL
- FORTRAN
- ALC
- JCL
- AUTOCODE
- PL/1
- PL/I
- BAL
- RPG
- COBOL
- USER180
- COBOL-72
- USER780
- DATA

Acceptable values for CA Librarian include the following:

- ASM
- JCL
- COB
- PLF
- DAT
- PLI
- FOR

- RPG
- FRG
- TXT
- GIS
- VSB
- GOF

LVLS TO CONSOL

Indicates the number of deltas to consolidate when the number of physical change levels reaches the figure in the CONSOL AT LVL field. Default is 50. This value must be zero if AUTO CONSOL=N, and cannot be greater than the value in the CONSOL AT LVL field.

When moving or transferring with history, CA Endevor SCM consolidates a number of levels equal to:

- The number in this field.
- The number of levels needed to reach the value in the CONSOL AT LVL field.

For example, if the value in this field is 30, and the value in the CONSOL AT LVL field is 70, at level 71 CA Endevor SCM consolidates the oldest 30 deltas into a single consolidation level (level 01).

Note: If age level retention is activated at the system level, the AUTO RETENT field appears in place of the AUTO CONSOL field and the CONSOL AT LVL and LVLS TO CONSOL fields do not appear.

REG SEV (I/W/C/E)

Determines severity of the error message issued when CA Endevor SCM detects regression. Valid values:

I— Informational message.

W— Warning message.

C— Default. Critical message.

E— Fatal message.

DATA FORMAT (T/B)

Indicates the element classification type that is used for file conversion when transferring files between CA Endevor SCM and CA CMEW, the Eclipse-Based UI, or a user-written Web Services client program. Acceptable values are the following:

B— Binary. An element exchanged between CA Endevor SCM and CA CMEW, the Eclipse-Based UI, or a user-written Web Services client program is not modified. This applies in both directions. B is required for any elements written in a double-byte character set (DBCS) or Unicode.

T— Text. Character set conversion is applied to an element. This is the default.

For more information about defining element types associated with files exchanged between CA Endevor SCM and CA CMEW, see the *CA CM Enterprise Workbench User Guide*.

HFS RECFM (COMP/CR/CRLF/CRNL/F/LF/NL/V)

Identifies the record delimiter used in a HFS file. A record delimiter is necessary due to the nature of HFS files. HFS files contain one large data stream. Therefore, a delimiter is used to identify individual records within that data stream.

This is also true for elements associated with a type defined as text (Data Format=T). When the element is transferred from CA Endevor SCM to CA CMEW or any other client application that uses CA Endevor SCM WebServices. These programs recognize and append the delimiter for the exchanged records. However, these delimiters are then translated from EBCDIC to the target code page. When the element is transferred to CA Endevor SCM, all delimiters are removed before translation. The one exception is where two delimiters appear consecutively, in which case a single space is inserted to avoid creating zero length records.

If a delimiter is not specified, the system defaults to NL.

Note: The change of a delimiter for types with existing elements may make these elements unusable for Retrieve actions.

Acceptable delimiter values are the following:

COMP— Variable length records compressed by CA Endevor SCM.

CR— Carriage return. The hex value is x'0D' (and also x'0D' in ASCII).

CRLF— Carriage return \ line feed. The hex value is x'0D25' (or x'0D0A' in ASCII). This is the default delimiter used on the Windows platform.

CRNL— Carriage return \ new line. The hex value is x'0D15' (or x'0D0A' in ASCII).

Note: CRNL is sometimes referred to as *CRLF*. However, CRNL should not be confused with the HFS RECFM value CRLF, which is the EBCDIC version.

F— Fixed Length. When the type is specified as fixed length, CA Endevor SCM does not expect or use any delimiters found in the file. This option should also be used for *Stream data* especially in binary elements, to allow the native file system (ASCII or EBCDIC) delimiters to remain unchanged.

LF— Line feed. The hex value is x'25' (or x'0A' in ASCII).

NL— Default. New line character. The hex value is x'15' (or x'0A' in ASCII). This is the delimiter used by the OEDIT and OBROWSE editors. It is also the default delimiter on Unix platforms and is supported in several Windows editor environments, making it a good choice for interoperability.

V— Variable. The first four bytes of the record contain the RDW (record descriptor word). The RDW contains the length of the entire record, including the RDW.

Note: For more information about exchanging HFS files between CA Endevor SCM and CA CMEW, see the appendix "Long Name and HFS Support."

FILE EXT

Indicates the 0- to 8-character file extension to be used by the CA CMEW for elements of this type. The extension can contain mixed case characters of a-z, A-Z and 0-9, or all blanks. Embedded blanks are not supported.

In the Eclipse-Based UI, the File ext (if specified) is used in the project view to help distinguish between elements with the same name, but different types. If the file extension is blank, then the CA Endevor SCM type name appears instead.

The file ext is also used when Creating elements, or using the Team, Add to CA Endevor SCM context menu, to filter the list of types so that only types that match the file extension are listed. In the case that no file extensions match, all CA Endevor SCM types are listed and the user may choose the correct type and processor group from the list.

Note: In the Eclipse-Based UI, Team, add to CA Endevor SCM context menu, when a file is added to CA Endevor SCM, the element name defaults to the file name minus the extension.

AUTO RETENT

(Display only.) This field is only displayed if the change levels are managed by age as defined on the system definition. When auto age level retention is set at the system level, change levels for all types in that system expire after the number of months specified on the system definition. For reverse (image) deltas, change levels that expire are dropped. For forward deltas, change levels that expire are incorporated into the base. When auto age level retention is turned off on the system definition, the auto consolidation options on the type definition apply. If auto age level retention is activated at the system level, the AUTO RETENT field appears in place of the AUTO CONSOL field. If this option is not activated at the system level, the AUTO CONSOL field appears. The auto age level retention feature can only be turned on or off on the system definition. Removal of expired levels only occurs after an element reaches the end of its map route.

Y— Display levels are managed by age.

Note: If the AUTO RETENT field is displayed, it replaces the AUTO CONSOL field and the CONSOL AT LVLS and LVSL TO CONSOL fields do not appear.

Component List Options

The component list base and delta members are stored in the delta library defined in the type definition. The following information describes the Component List Options fields:

Fwd/Rev Delta

Specifies delta storage format for component list information. Acceptable values are the following:

R

Reverse delta format.

F

Default. Forward delta format.

Auto Consol

Indicates whether CA Endevor SCM is to consolidate component list delta levels automatically in conjunction with Consol at Lvl and Lvls to Consol. Acceptable values are:

Y

Consolidate automatically, when physical Consol at Lvl is reached. For example, if Consol at Lvl is set to 96, and Lvls to Consol is set to 50, when the 96th physical component list delta levels are reached, CA Endevor SCM will automatically consolidate the first 50 change levels into a single level, and adjust the level number accordingly.

N

Default. Do not consolidate automatically. The LVLS TO CONSOL field must be set to 0.

Note: If auto age level retention for component is activated at the system level, the AUTO CONSOL field does not appear. It is replaced by the AUTO RETENT field and the CONSOL AT LVL and LVLS TO CONSOL fields do not appear.

Auto Retent

(Display only.) This field is only displayed if the change levels are managed by aged as defined on the system definition. When auto age level retention is set at the system level, change levels for all types in that system expire after the number of months specified on the system definition. For reverse (image) deltas, change levels that expire are dropped. For forward deltas, change levels that expire are incorporated into the base. When auto age level retention is turned off on the system definition, the auto consolidation options on the type definition apply. If auto age level retention is activated at the system level, the AUTO RETENT field appears in place of the AUTO CONSOL field. If this option is not activated at the system level, the AUTO CONSOL field appears. The auto age level retention feature can only be turned on or off on the system definition. Removal of expired levels only occurs after an element reaches its end of route location.

Y

Display levels are managed by age.

Note: If the AUTO RETENT field is displayed, it replaces the AUTO CONSOL field and the CONSOL AT LVLS and LVSL TO CONSOL fields do not appear.

Consol at Lvl

Specifies the number of physical component list delta levels at which CA Endevor SCM will perform automatic consolidation.

Default: 96.

For example, if the value in this field is 70, CA Endevor SCM consolidates levels when component list delta levels 71 is reached.

Note: If auto age level retention for components is activated at the system level, the AUTO RETENT field appears in place of the AUTO CONSOL field and the CONSOL AT LVL and LVLS TO CONSOL fields do not appear.

Lvls to Consol

Indicates the number of deltas to consolidate when the number of physical levels reaches the figure in the CONSOL AT LVL field. Default is **50**. This value must be zero if AUTO CONSOL=**N**, and cannot be greater than the value in the CONSOL AT LVL field.

When moving or transferring with history, CA Endevor SCM consolidates a number of levels equal to:

- The number in this field.
- The number of levels needed to reach the value in the CONSOL AT LVL field.

For example, if the value in this field is 30, and the value in the CONSOL AT LVL field is 70, at level 71 CA Endevor SCM consolidates the oldest 30 deltas into a single consolidation level (level 01).

Note: If auto age level retention for components is activated at the system level, the AUTO RETENT field appears in place of the AUTO CONSOL field and the CONSOL AT LVL and LVLS TO CONSOL fields do not appear.

Libraries

These library names can be specified using symbolics. The following information describes the Libraries fields.

Note: For more information about the available symbolics, see Using Symbolics to Define Base and Delta Libraries (see page 117).

Base/Image Library (Required)

Name of the base library for the type. Can be PDS, PDSE, CA Panvalet, CA Librarian, ELIB, or USS.

Delta Library (Required)

Name of the delta library for the type. Can be PDS, PDSE, CA Panvalet, CA Librarian, or ELIB.

Include Library (Optional)

Name of the PDS, PDSE, ELIB, CA Panvalet, or CA Librarian INCLUDE library for the type. If specified, members can be included and expanded from this library.

Note: If an ELIB is specified as an INCLUDE library, it must be reverse delta and have a name that is not encrypted.

Source Library (Optional)

Data set name of source output library.

Expand Includes (Optional)

Indicates whether Include statements are expanded when the element is written to the source output library. Acceptable values are:

Y

Expand Include statements.

N

Do not expand Include statements.

Type Naming Conventions

Consider using generic type names, such as COBOL. You can then create as many processor groups as you need to handle the variations within each type. For instance, for type COBOL, you could create processor groups to tell CA Endevor SCM what type of COBOL must be processed.

Note: For more information about processor groups, see the *Extended Processors Guide*.

Suggested Naming Standards

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

ASEMBLER	EASYTREV	MARKV	SORTCNTL
BASIC	FORTRAN	NETRON	SPECS
CLIST	JCL	PLI	TABLES
COBOL	LINKCARD	REPORTS	TRNSFORM
COPYBOOK	MACRO	RPG	UTILITY

Element Storage Formats

Each time CA Endevor SCM moves or transfers an element from one location to another location source compare is performed, which results in a new delta level if changes are found. A delta level is a record of the changes to the base level of an element.

Element Types can be defined with one of several delta storage formats. Each storage format has a different way of managing the changes made between element levels. Each Type definition must specify one of the following storage formats:

- Reverse delta format (see page 115)
- Forward delta format (see page 115)
- Image delta format (see page 116)
- Log delta format (see page 116)

The element Type storage formats are compared in the following table:

Element Format	Current Level	Prior Levels	Element Base Level	Delta Level Files
Reverse delta	Complete image	To create a specific level prior to the current level, all delta files up to and including the specific prior level's delta are applied to the *element base* to remove all changes made at those levels.	Current level	Show changes between each level.
Forward delta	To create the current level, all the delta levels are applied to the element base.	To create a specific level prior to the current level, all delta files up to and including the specific prior level's delta are applied to the *element base* to add all changes made at those levels.	The image is as it was first added to CA Endevor SCM	Show changes between each level.
Image delta	Complete image	Each level is stored as a complete image showing the current state of the element.	Current level	Each delta file is a complete image, because changes between levels are not compared.
Log delta	Complete image	N/A	Current level	Each delta file contains change information, but not source code changes. Data in the Source Level Information area in an element browse and summary display comes from information stored on the change level delta records.

Reverse Delta Format

The *reverse delta format* stores the most recent version of the code, rebuilding prior versions by backing out individual changes from the current version. The current level of the element is a full image of the element. To create a previous level of the element, CA Endevor SCM applies the previous levels' delta files to the current level of the element. This process reverses, that is removes, the changes made to the current level since the previous level. Using reverse deltas lets you store the element base in standard PDS format (not encrypted, noncompressed).

The reverse delta storage format has the following requirements and benefits:

- Does not allow two elements with the same name to be stored in the same base library. Keep this in mind when planning your inventory and library structure.
- Requires separate base libraries by stage and type. Delta libraries can still be shared across stages in an environment.
- Uses two I/Os for writes, one for reads.
- Allows a regular PDS to be used to store element base and an ELIB data set to store deltas.
- Allows outside utilities to read base libraries directly. Examples of outside utilities include File Aid and compilers.
- Eliminates the need for CONWRITE in processors.
- Eliminates writes to source output libraries, which also saves DASD.

Forward Delta Format

The *forward delta format* stores a base version of code, and then builds current versions by applying changes made to the base. The element base is the same as the element when it is first added to CA Endevor SCM. When CA Endevor SCM processes elements stored in this format, it applies all delta levels to the base to create the current image of the element. To create a level prior to the current level, all delta files up to and including the prior level's delta are applied to the element base. If you select this format, CA Endevor SCM encrypts and compresses both the base and delta files.

With the forward delta format, the element base and subsequent deltas files are stored in encrypted/compressed format. This format has the following requirements and benefits:

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

Image Delta Format

The *image delta format* is for elements for which source compare is impossible or unnecessary, such as USS binary executables. In other cases, the CA Endevor SCM compare algorithm can require a significant amount of time to complete.

The image delta format stores each element level as a full image of the element. Changes between levels are not compared. Only display or print requests for Browse, Change Summary, and Master requests are allowed for image elements. Requests for Changes and History display are not supported.

However, the full-image delta format enables you to do the following:

- Track changes by date.
- Track changes by user.
- Associate comments to the change.
- Associate a CCID to the change.
- Retrieve prior versions of the element.

Log Delta Format

The *log delta format* is for very large data files, HFS files, and binary files (for example, JAR, WAR, EAR, DOC, PPT, and XLS files). For such files, source management is not needed, but an activity log can be helpful. The log delta format stores only the last level of the element and stores it as a full image of the element. Delta files contain only change activity information by level. Source changes by level are not kept. Consequently, you cannot view prior source levels or display source change or history.

When defining the log delta format for an element Type definition, you can specify whether CA Endevor SCM is to compress the base and encrypt the name. The regression percent specification must be set to zero.

Element components do not use the log delta format; components are managed by forward and reverse delta formats.

Restrictions apply to converting elements to different formats. However, elements with forward, reverse, or image delta formats can be changed to log delta format. This change can occur when elements are restored (using the Restore action) or transferred from an unload or archived file. You can use the Restore action to restore elements with delta types other than log delta from an archive file in to a CA Endevor SCM log delta Type location. The Reload utility can be used to similar effect.

The following restrictions apply to the use of the log delta format:

- An element delta type other than log cannot be moved with or without history, into a log delta Type location. This restriction also applies to transfers using the CA Endevor SCM to CA Endevor SCM Transfer action.
- After a log delta element is created, it cannot be moved into another location that is not managed by a log delta Type. This restriction also applies to transfers using the CA Endevor SCM to CA Endevor SCM Transfer action.
- If sourced elements exist, the delta storage format defined for an element Type cannot be changed *from* log delta format to another format.
- If sourced elements exist, the delta storage format defined for an element Type cannot be changed *to* log definition from another format. This restriction prevents accidental source history delta deletion. For example, assume that forward delta elements exist at a location and the Type definition changes to log. Then, the next altering source management activity would convert these elements to log deltas. This action would purge all source history.

Note: For more information, see How to Convert Elements to Log Delta Format (see page 273).

Using Symbolics to Define Base and Delta Libraries

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

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

Organize libraries by	Using this symbolic	or this alias
Type	&C1ELTYPE	&C1TY

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

For example, the following library specification would build a library called SRCLIB for each subsystem/stage combination within a given system.

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

How to Define an Element Type for Text Files

To enable Web Services to transfer text files using attachments, Type definitions for text files must be properly configured in CA Endevor SCM. CA Endevor SCM defines a text file as one that has line delimiters and requires character translation. You can store text files on the mainframe and browse them in CA Endevor SCM. To define a text Type, complete the following steps:

1. Define the base and delta libraries using CA Endevor SCM rules for the specific file type. For more information, see the *CA Endevor Software Change Manager Administration Guide*.
2. Set TYPE DEFINITION panel fields to the following values to define a CA Endevor SCM Type record for text data.
    a. Set the FWD/REV/IMG DELTA field to F for forward or R for reverse to specify the delta storage format.
    b. Set the COMPRESS BASE/ENCRYPT NAME field to N for element names that do not exceed eight characters and are uppercase. For other element names, set this field to Y. If set to Y, the base library must be HFS.
    c. Set the COMPARE FROM field to a value appropriate for the type of data. For example, COBOL can be 7.
    d. Set the SOURCE LENGTH and COMPARE TO fields to values appropriate for the type of data. For example, COBOL SOURCE LENGTH can be 80 and COMPARE TO can be 72.
    e. Set the HFS RECFM field to CRLF. This is the line feed character for Microsoft Windows files. If you do not specify CRLF for text types accessed using local file system support, you will get unexpected results in Windows-based editors.

f. Set the DATA FORMAT field to T for text. This triggers conversion from EBCDIC to local file system encoding.

g. Set the FILE EXT field to the file name extension you want the file type to have on the local machine (cob, jcl, and so on) or leave it blank. FILE EXT is used when you are adding files from or retrieving files to a local directory using Web Services.

How to Define an Element Type for Binary Files

To define an element Type for binary files (for example, WAR, EAR, JAR DOC, PPT, and XLS), use the following steps:

1. Define the Base library—The base library can be PDS, PDSE, ELIB, CA Panvalet, or CA Librarian.
2. Define the Delta library—The delta library can be PDS, PDSE, or ELIB. Define the delta library as follows:
   - The delta PDS, PDSE, or ELIB record format must be variable blocked.
   - The delta library record length and the Type definition source length can be any value (for example, 259, 6024 or 27984). However, define the maximum delta record length about twice the source length. We recommend an LRECL value of 27,984. For example:

     ```
     DCB=(DSORG=PO,RECFM=VB,LRECL=27984,BLKSIZE=0)
     ```
3. Type definition— Set the TYPE DEFINITION to the following values to define the CA Endevor SCM Type record for the binary element data.
   - Set the Fwd/Rev/Img/Log Delta field to one of the following formats:
     - **I**— Image delta format. The image delta format stores the full image of the file. This format suppresses the comparison logic. Each update causes the entire file to be added as a new level. *Using the full image delta can cause the delta file to run out of space if you have very large binary files and set a high consolidation level.*
     - **L**— Log delta format. The log delta format stores only the last level of the element and stores it as a full image of the element. Delta files contain only change activity information by level. *Source changes by level are not kept. Consequently, you cannot view prior source levels or display source change or history.*

     Important: Before deciding which delta format to use, consider the benefits and limitations of both. The image delta format can use much space, but prior levels are retained. Whereas, the log delta format only keeps the current element, but does not keep prior levels. For more information about delta formats, see Element Storage Formats (see page 113).
   - Set the COMPARE FROM field to 1. This option specifies the position within each statement at which CA Endevor SCM begins comparing to identify changed statements (5 digits in the range 1-32,000).
   - Set the SOURCE LENGTH field to 13992. This option specifies the logical record length in the source statements. Although the maximum allowable value is 32000, the value must not exceed the maximum physical record length in the delta library definition.
   - Set the COMPARE TO field to 13992. This option specifies the position within each statement at which CA Endevor SCM stops comparing to identify changed statements.

- Set the HFS RECFM field to F to specify the fixed length. This option specifies the record delimiter used in a HFS file. A record delimiter is necessary due to the nature of HFS files. HFS files contain one large data stream; therefore, a delimiter is used to identify individual records within that data stream. This is also true for elements associated with a Type defined as Data Format=Text when the element is transferred between CA Endevor SCM and Web Services. Web Services recognizes and appends the delimiter for the exchanged records.
- Set the DATA FORMAT field to B for binary. If left blank, the value defaults to B. This option deactivates compression and prevents text character conversion from being invoked.
- Set the FILE EXT field to a valid file extension (doc, ppt, xls, jar, ear, and so on) or leave it blank. The file extension identifies the file type when you are adding from or retrieving to a local file directory using CA CMEW, the Eclipse-Based UI, or a Web Services user-written client program.
- Set the COMPRESS BASE/ENCRYPT NAME field to Y or N. This option specifies whether to encrypt and compress the base form of elements stored in reverse delta format.

Defining the Type Processing Sequence

You have the option of enforcing an element type processing sequence within an environment/system definition or through the definition of a single file which enforces an element type processing sequence at the site level. With Global Type Sequencing, all elements across all environments follow the same processing sequence.

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

Before using this option, you must first define at least two element types for the system.

Note: For more information, see Defining Types (see page 100).

Each time you add or delete a type thereafter, use option **7** again to redefine the relative order of processing for the system. When a new type is added, it defaults to being processed last.

Note: For more information, see Global Type Sequencing.

Define the Type Processing Sequence

Use this procedure to define the type processing sequence.

1. Select option **7** on the Environment Options panel and press Enter.

   The Type Sequence Request panel appears.

2. Select an option (Blank or U).
3. Enter an environment name, if different from the displayed name.
4. Enter a system name or mask.
5. Enter a stage ID.
6. Press Enter.
   - If CA Endevor SCM displays a System Selection List, select a system, press Enter, and then proceed to Step 7.
   - If CA Endevor SCM displays a Type Processing Sequence Definition panel, proceed to Step 7.
7. To reorder a type, change its sequence number with:
   - A number smaller than the number of the type you want it to precede.
   - A number greater than the number of the type you want it to follow.
8. Press Enter to save the changes.

Type Processing Sequence Panel Fields

The following information describes the fields on the Type Processing Sequence panel. All fields except the Relative Processing Sequence field are display-only.

Current Env

Name of the current environment.

Stage ID

ID of the current stage.

System

Name of the current system.

Next Env

Name of the environment at the next map location.

Next Stage ID

ID of the next stage at the next map location.

Next System

Name of the next system at the next map location.

Updated

Identifies the date, time, and user ID of the last user to update the processing sequence definition. When creating a new type processing sequence, this field is blank.

Relative Processing Sequence

A three-digit number that defines the relative processing order for processors executed in batch, for the type displayed to the right. Type over this number as appropriate to change the type processing. For example, assume you have these processors set up in this sequence:

- 010 CLISTS
- 020 JCL
- 030 PROC
- 040 COPYBOOK

You decide that you want to run JCL after PROC. You can reorder the processors by typing over the appropriate sequence numbers; in this example, you need renumber only JCL and PROC, as follows:

- 025 JCL
- 015 PROC

Note: Other sequence numbers do not need adjusting. You can use any value from **1- 9** to indicate a changing sequence; that is, instead of assigning 025 to JCL, you could use 021 or 029. The value you enter must be unique. If you enter a duplicate number, you receive an error message.

When you press Enter, CA Endevor SCM redisplays the panel to reflect the new order, with the numbers adjusted to start with 010 and increment by 10. In the previous example, the panel would reflect the following sequence:

- 010 CLISTS
- 020 PROC
- 030 JCL
- 040 COPYBOOK

Type

Name of the type whose relative processing sequence is shown to the left.

Description

Description of the type.

Updating Type Data Set Definitions

This section describes how to view and change the name of one or more data sets defined for a system.

The Type Data Set Request Panel

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

When you select option **8**, and press Enter, CA Endevor SCM displays the Type Data Set Request panel.

Define the Type Data Set Definitions

Use this procedure to view or change the name of one or more data sets defined for a system.

1. Select option **8** on the Environment Options Menu.

   The Type Data Set Request panel appears.

2. Select an option (**Blank** or **U**).
3. Enter an environment name, if different from the displayed name.
4. Enter a system name or mask.
5. Enter a stage ID.
6. Press Enter.
   - If CA Endevor SCM displays a System Selection List, select a system, press Enter and proceed to Step 7.
   - If CA Endevor SCM displays a Type Data Set Definition panel, proceed to Step 7.
7. Type or change information as necessary on the Type Data Set Definition panel, and press Enter to save the changes.

The Type Data Set Panel

CA Endevor SCM displays the Type Data Set Panel after you select a system. This panel lists the data sets defined for the element types within that system, in order as they were defined. To change a library, type over the data set name, and press Enter.

Type Data Set Panel Fields

The following information describes the fields on the Type Data Set panel. All fields except the Data Set Name field are display-only.

Current Env

Name of the current environment.

(Current) Stage ID

ID of the current stage.

(Current) System

Name of the current system.

Next Env

Name of the environment at the next map location.

(Next) Stage ID

ID of the next stage at the next map location.

(Next) System

Name of the next system at the next map location.

Data Set Name

Name of a library used by an element type defined within this system. For each element type, the list includes the base library, delta library, Include library, and source output library, as appropriate to the type definition. The libraries are displayed in order as they were defined to CA Endevor SCM. Type over this field to change one or more library names.

Last Modified by

The Last Modified By fields identify the user ID of the last user to update the type data set as well as the date and time the update took place.

Type

Type of library:

PO

Partitioned Data Set

EL

ELIB

PV

CA Panvalet

LB

CA Librarian

If the data set name was created using symbolics, CA Endevor SCM displays "??" in this field.

Msg

The following messages can appear in the Msg field:

***NCAT**

Indicates that the data set is not catalogued.

***ISYM**

An invalid CA Endevor SCM symbol was specified in the data set name.

***SERR**

A symbol parsing error occurred.

***IDSN**

A new data set name specified that contains invalid characters.

***NMNT**

The volume on which the data set is catalogued is not accessible (therefore the data set organization cannot be derived).

***MVOL**

The data set is a multi-volume data set.

***CTIO**

An I/O error occurred while trying to locate the data set in the system catalog.

***ERR**

An unexpected error occurred.

***UPDT**

The data set update was processed and completed.

Displaying Environment Information

This section describes how to display environment information.

The Environment Information Panel

To display the environment definitions defined during installation, select option **E** on the Environment Options Menu, and press Enter. CA Endevor SCM displays the Environment Information panel. The values displayed on this panel are obtained from your current Defaults Table.

To return to the Environment Options Menu, press End.

Environment Information Panel Fields

The following information describes the fields on the Environment Information panel. All fields except the CURRENT ENVIRONMENT field are display-only.

Current Environment

Name of the current environment. To display the details for another environment, enter the environment's name in this field, and press Enter.

Title

Descriptive title for the current environment.

Next Environment

Name of the next environment in the map.

User Security Table

(Applicable for native security only) Name of the User Security Table currently in use.

Resource Security Table

(Applicable for native security only) Name of the Resource Security Table currently in use.

Journal

Applicable only to users of CA Endevor SCM's Point in Time Recovery facility. Name of the journal file used by this facility.

SMF Activity

Indicates whether the SMF facility is active for this environment: **Y** or **N**

SMF Security

Indicates whether SMF security has been turned on for this environment: **Y** or **N**

MVS/DB Bridge

Indicates whether the CA Endevor SCM to CA Endevor/DB for IDMS Bridge is used in this particular environment: **Y** or **N.**

Chapter 4: Defining Maps

This section contains the following topics:

Maps

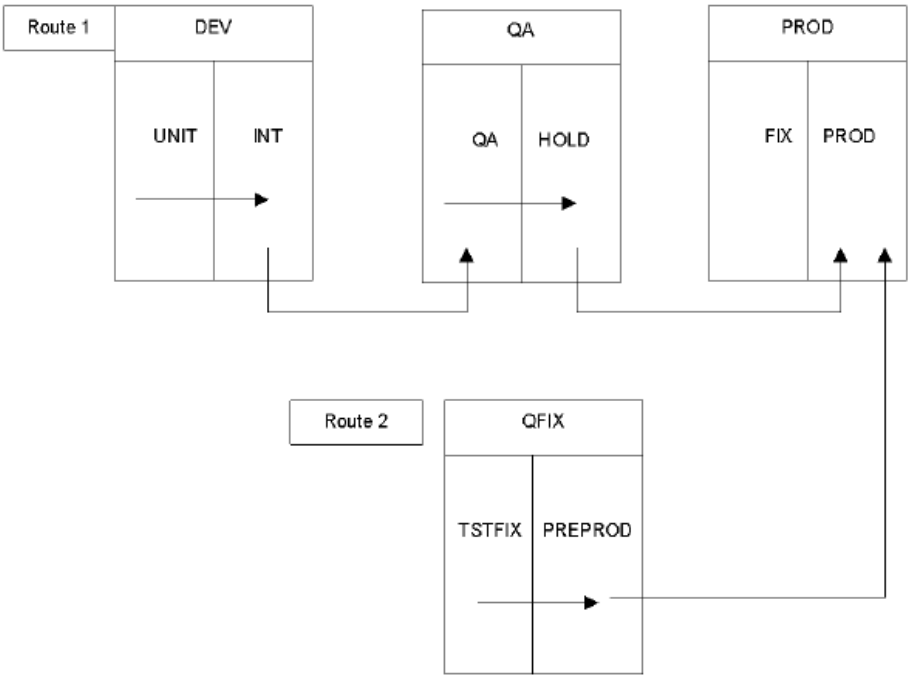
Part of implementing CA Endevor SCM is to define the stages in the software lifecycles at your site, then organize these stages into environments. This section of this manual illustrates this process.

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

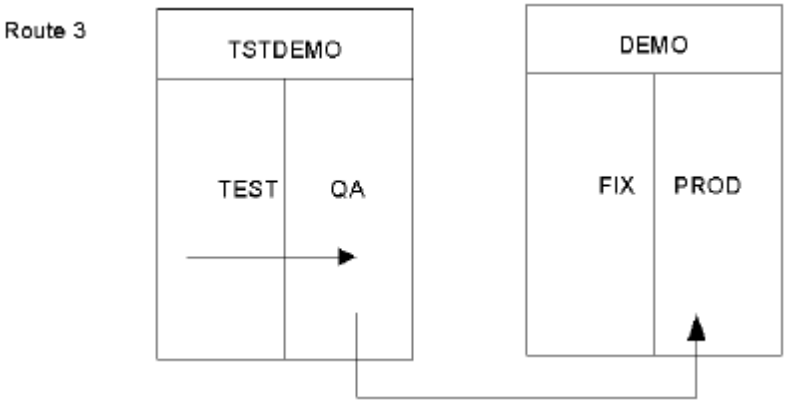
CA Endevor SCM uses these routes to automatically add, display, retrieve, move, and generate inventory in a given lifecycle.

Consider the following examples:

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



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



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

How to Define a Map

You define a map by:

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

Establish Routes in the Defaults Table

To define a route, add the following statement to one or more C1DEFLTS TYPE=ENVRNMNT sections of the Defaults Table:

```
NEXTENV=(environment-name, stage-id)
```

The following information explains the variables in the previous statement:

environment-name

The name of the next environment in the route.

stage-id

The one-character identifier of the first stage in that environment that is on the route. The stage ID is optional, and if included, must be defined in the Defaults Table.

For example, to define Route 1 in the preceding section, add the following to the C1DEFLTS TYPE=ENVRNMNT section:

NEXTENV=QA

For environment DEV.

NEXTENV=(PROD,P)

For environment QA.

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

Mapping Inventory Classifications

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

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

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

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

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

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

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

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

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

Note: For more information, see Defining Inventory Structures (see page 73) and the *Extended Processors Guide*.

Design Strategies and Guidelines

This section describes the strategies and guidelines for defining routes.

Routes

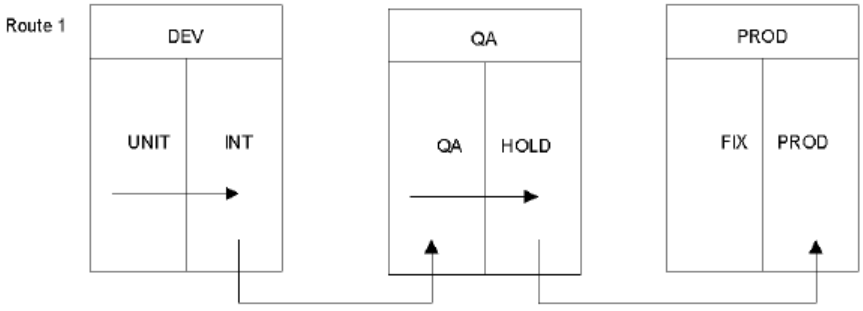
The routes that you develop for your site can have a significant impact on your success in using CA Endevor SCM. When defining routes, consider the following information.

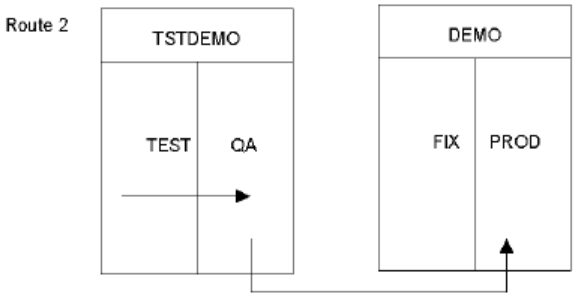
- System and subsystem names can change only when going from one environment to another.
- Processor group names can change when going between stages or between environments.

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

Stand-alone Routes

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



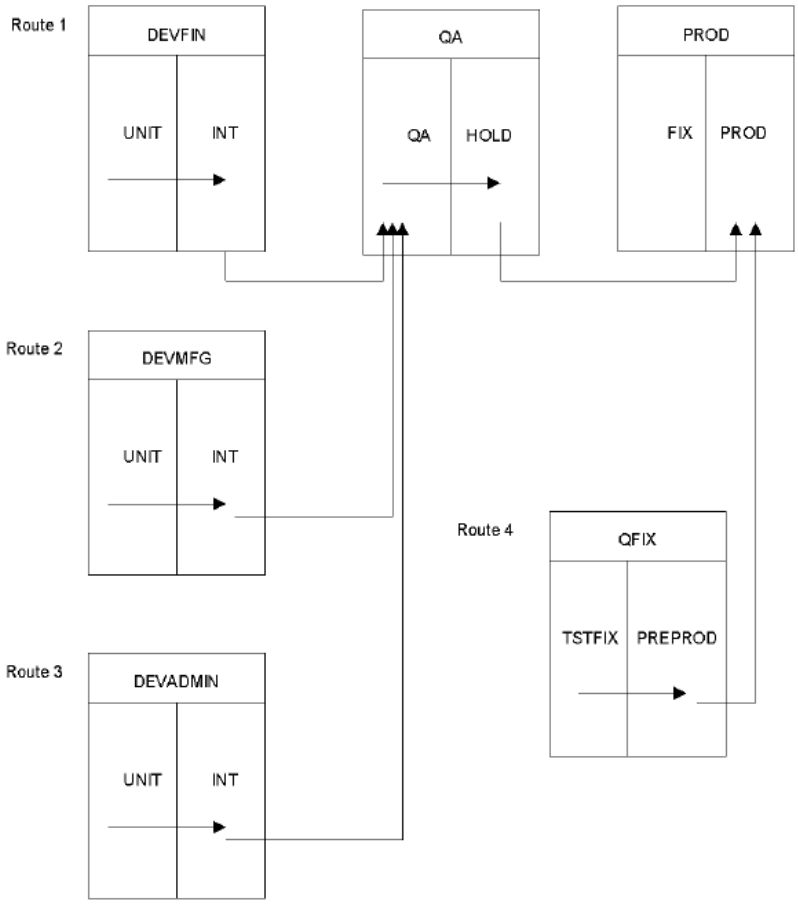


Each route is defined separately as follows:

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

Converging Routes

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



There are four routes at this site:

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

Converging Systems within a Route

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

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

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

The administrator defines an environment map by adding the following lines:

NEXTENV=QA

Add this line to the C1DEFLTS TYPE=ENVRNMNT section for environment DEV.

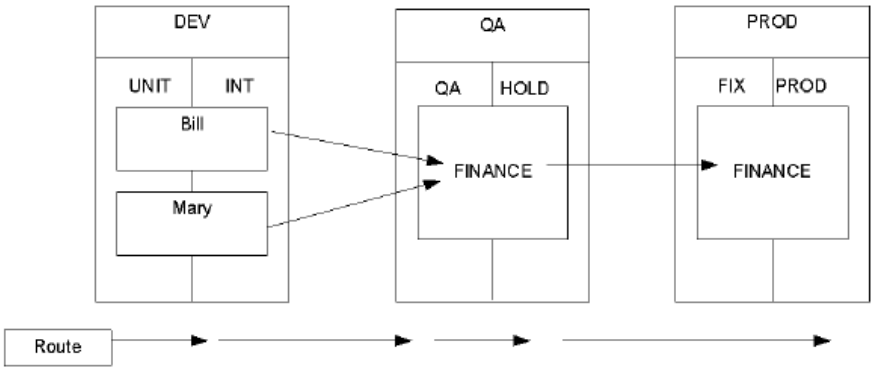
NEXTENV=(PROD,P)

Add this line to the C1DEFLTS TYPE=ENVRNMNT section for environment QA.

The administrator completes the NEXT SYSTEM, NEXT SUBSYTEM, and NEXT GROUP fields on the respective definition panels as follows:

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

The following diagram represents the route:



Implementation Checklist

Use the following steps for reference when defining routes.

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

   Note: You can also define your maps using batch SCL commands. For more information, see the *SCL Reference Guide*.
4. Run CONRPT07, System Definition Profile, to verify the routes are set correctly. In addition, run CONRPT07 whenever you change a route to verify that you have made the modifications correctly.

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

Chapter 5: Using Element Registration

This section contains the following topics:

Element Registration

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

When you define a system, CA Endevor SCM provides two options that enable you to allow or disallow duplicate element names. One option enables you to control the use of duplicate element names across subsystems within the system. The other option enables you to control the use of duplicate element names at the processor group level within the system. Both ways of controlling duplicate element names can be made to apply across systems.

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

Controlling Duplicate Element Names at the System and Subsystem Level

The Element Registration option enables you to control whether duplicate element names are allowed across subsystems within a system. During action processing, when the subsystem associated with the element is validated, CA Endevor SCM checks the option to see if duplicate element names are allowed.

By default, element name registration is defined at the system level on the system definition. You can also specify the element name registration options you want to apply to element name registration across systems. This is an optional feature you can activate in the optional features table ENCOPTBL by turning on the option REGISTER_ACROSS_SYSTEMS. If this option is turned on, CA Endevor SCM ignores the element name registration setting on the system definitions.

There are two sets of parameters that you can use to control duplicate element names at the system and subsystem level:

- System Definition Parameters

  The System Definition parameters Duplicate Element Name Check and Msg Severity Lvl, govern this option. You can specify the following parameter values:

  E (Error)

  The same element name exists within another subsystem under the same system; the action is terminated and an error message is issued.

  C (Caution)

  The same element name exists within another subsystem under the same system; the action is performed and a caution message is issued.

  W (Warning)

  The same element name exists within another subsystem under the same system; the action is performed and a warning message is issued.

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

- Element Registration across Systems

  If you turn on the optional feature REGISTER_ACROSS_SYSTEMS in ENCOPTBL, then CA Endevor SCM issues a message if an element is added or created and another element with the same name exists in another unmapped system/subsystem. If the option is set to E as follows: REGISTER_ACROSS_SYSTEMS=(ON,E), then you *cannot* create an element with the same name unless it is in the map of an existing element. A setting of C or W allows you to create a duplicate element, but a warning or caution message is issued.

  Regardless of whether the option is set to E, C, or W, element name registration restricts the use of the same element name no matter its type. With this option turned off, elements with the same name can only exist in the same system/subsystem or in another system (no matter the subsystem). With this option turned on, elements with the same element-name (no matter the type) can only exist if one maps to the other.

  When this option is turned on, CA Endevor SCM ignores all option settings for element name registration at the system definitions. The option-severity set in ENCOPTBL will rule over all systems in all the environments no matter what the element name registration option settings in the system definitions contain. If the REGISTER_ACROSS_SYSTEMS option is set to OFF, or is not activated in ENCOPTBL, then the settings on the system definition are used to determine what rules apply. This option has no impact on the way element registration works at the processor group level. You can continue to define elements with the same name in the same system/subsystem if the Duplicate Proc O/P Type check settings allow this.

Controlling Duplicate Element Names at the Processor Group Level

Element Registration at the processor group level lets you control whether two elements with the same name, but with different element Types, can exist in the same System when both elements are associated to processors groups with the same processor output Type. The restriction can be set to all Systems or within a specific System. The purpose of the option is to prevent unintentional overlays of processor output library members that have the same name.

During action processing, if the element already exists at the target location within the same System under a different Type with the same output Type, the action is terminated or a warning message issued. This also occurs if an element with the same name and processor output Type, but different Type, already exists in a different System, provided the optional feature to check across Systems is enabled in ENCOPTBL.

Optionally, the search can extend to the Subsystem level. In this case, the searches for duplicate element names at the processor group level extends across Subsystems defined to a specific System. Specific Subsystems can be excluded from the search.

How to Enable Element Registration for Processor Groups

The Element Registration feature for processor groups can be enabled in a System definition to apply to all Subsystems within the System. To exclude any Subsystem, the Subsystem definition must be set to exclude it from the search. The administrator can set the options for the System and Subsystem definitions in foreground or batch. For more information about Batch Administration, see Define System Syntax and Define Subsystem Syntax in the *SCL Reference Guide*.

To enable Element Registration at the processor group level, perform the following steps:

1. Set the following parameters on the System Definition panel:

   DUP PROC O/P TYP=Y – Checks element names across Types and processors groups for the same processor output Type. A conflict occurs if you attempt to add or create an element and an element with the same name and processor output Type exists in the same System, with a different Type .

   ACROSS SBS=Y – (Optional) Extends the check for duplicate processor output Types to all the Subsystems defined to this System. The check compares the processor output Type of the current action against all processor output Types of all same named elements in the different Subsystems of the same System. If any are equal, the current action fails with a registration error.

MSG LEV - Specifies the message severity level when the check finds a duplicate. Valid values are:

W (Warning) – The action is performed and a warning message is issued.

C (Caution) – The action is performed and a caution message is issued.

E (Error) – The action is *not* performed and an error message is issued. The default.

This activates Element Registration for processor groups for the System you are updating on the System Definition panel. If you specify ACROSS SBS=Y, the check will extend to the Subsystems defined to this System.

2. (Optional) If the ACROSS SBS=Y option is set on the System definition, you can exclude specific Subsystems from the check. To exclude a Subsystem from the check, set the following option on the Subsystem Definition panel:

   EXCLUDE FROM DUPLICATE ELEMENT PROC O/P TYPE CHECK=Y – Excludes this Subsystem from the check.

3. (Optional) To activate Element Registration across *all* Systems at the processor group level, the following parameter must be set in the Optional Features table (ENCOPTBL):

   ```
   ENHOPT ELM_REG_CHK_OUTPTYPE_ACROSS_SYSTEMS=ON
   ```

Note: For processor groups, you can define an output Type or use the default output Type. The default processor output Type is a concatenation of the element Type and the processor group name. Using the default can help you avoid duplicates.

Defining the Output Type

After enabling the processor group option, you need to define the output type. The default output type is a concatenation of the element type and processor group names. Using the default value ensures there are no registration conflicts. Alternatively, you can define the output type using the Processor Group Definition panel. The output type field, PROCESSOR O/P TYPE, is 16-characters long.

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

Note: For more information about processor groups, see the *Extended Processors Guide.*

Chapter 6: Using Optional Features

This section contains the following topics:

The Optional Feature Table ENCOPTBL

The optional feature table (OFT) supplied with CA Endevor SCM provides you with a simple, easy-to-use mechanism to customize the product for use at your site. This facility enables you to configure your implementation of CA Endevor SCM by specifying which features you want to activate. After modifying the OFT source, you must assemble and link the table, placing the output into the CA Endevor SCM execution library.

The optional feature table embedded in ENCOPTNS provides a description of each option along with detailed information on parameter values and downstream effects.

The OFT source, as distributed, contains each option already coded, but inactive. To activate an entry, remove the asterisk (*) in column 1 and verify that the second parameter, if required, contains the appropriate value. Use sample JCL BC1JTABL to assemble and link ENCOPTBL to your *iprfx.iqual.CSIQAUTU* outside of SMP/E or use an SMP/E USERMOD to accomplish this. BC1JTABL is supplied in the installation library *iprfx.iqual.CSIQJCL*.

The optional feature table is provided with the CA Endevor SCM installation files as member ENCOPTBL in the installation TABLES library (iprfx.iqual.CSIQSRC).

The CCID Definition Data Set

If you want to use CCID validation, you must build a CCID definition data set. Before you can build the data set, though, you need to specify a CCID validation data set name in the Defaults Table, using the CIPODSN parameter.

Note: For more information about the CIPODSN parameter, see The TYPE=MAIN Macro (see page 208).

How to Install the CCID Definition Data Set

The following are the three steps involved in installing the CCID definition data set:

1. Allocate the data set.
2. Initialize the data set.
3. Add the data set name to the CA Endevor SCM Defaults Table.

Allocate the Data Set

Allocate the CCID validation data set using the standard ISPF data set utilities-option **3.2** from the ISPF/PDF Primary Option Menu. The data set must have the following attributes:

LRECL=

80

RECFM=

FB

DSORG=

PS

BLKSIZE=

Any multiple of 80

The data set name you use must be the same as that specified for the CIPODSN parameter in the Defaults Table. The data set name can be any valid name that conforms to your site's naming conventions.

How to Install the ++CONTROL Password Interface

The three steps involved in installing the ++CONTROL password interface include the following:

1. Edit member C1PTCNTL.
2. Assemble and link C1PTCNTL.
3. Copy the output into the CA Endevor SCM authorized library.

Edit C1PTCNTL

To install the ++CONTROL password interface, you need to provide the CA Panvalet library data set name and password for each library that is password-protected. Use member C1PTLCNTL, which is supplied in the installation *iprfx.iqual.CSIQSRC* library, to do this. C1PTLCNTL contains the instructions you need to define each library and password. Make sure that you provide values that are proper for your site for all the PANPRFX, PANQUAL, and PANLIB qualifiers within this module.

Assemble and Link-Edit C1PTCNTL

Use sample JCL BC1JTABL to assemble and link-edit C1PTCNTL to your iprfx.iqual.CSIQAUTU outside of SMP/E or use an SMP/E USERMOD to accomplish this. BC1JTABL is provided in the installation library iprfx.iqual.CSIQJCL.

Initialize the Data Set

You can initialize the CCID validation data set by copying a template that is delivered as part of CA Endevor SCM. Because CA Endevor SCM requires information to be placed in certain columns, it is recommended that you copy member SAMPCIPO into the data set allocated in Step 1. By using this member as a template, you can ensure that the fields in the data set are positioned in the correct columns.

Use the standard ISPF copy utility-option **3.3** on the ISPF/PDF Primary Option Menu-to copy SAMPCIPO from the sample JCL library (iprfx.iqual.CSIQJCL, created during Step 2 of the base installation process) into the data set.

Use the ISPF edit service to tailor the data set to meet the requirements of your site.

Add the Data Set Name to the Defaults Table

After you have allocated and initialized the CCID validation data set, you need to add the data set name to the Defaults Table. To do this, add the valid data set name to the CIPODSN parameter, as follows:

```
Col 1     Col 16                                          Col 72

          CIDEFLTS TYPE=MAIN,                             X
          .
          .
          .
          CIPODSN=,           CCID VALIDATION DATASET NAME   X
          .
          .
          .
```

Note: For more information about editing the table, see The Defaults Table (see page 207).

Do not update the CIPODSN parameter in the Defaults Table before allocating the CCID validation data set. If CIPODSN contains a non-null entry, CA Endevor SCM assumes that the entry is a data set name and looks for that data set during initialization. If CA Endevor SCM cannot find the data set, it generates an error message. Therefore, before using CA Endevor SCM, be sure that the CCID validation data set is allocated and defined appropriately.

The CA Panvalet Interface

If your site is (or will be) using CA Panvalet as a source library manager, you must link-edit the CA Panvalet access method (PAM) module into several of the CA Endevor SCM CA Panvalet Interface modules. Installing this interface also allows you to expand ++INCLUDE statements. These statements are expanded from either a CA Panvalet library or a PDS when the interface is enabled.

Note: Be sure that the CA Panvalet installation does not suppress level checking on the CA Panvalet UPDATE ALL command.

How to Link-Edit the Access Module

The following are the three steps involved in link-editing the CA Panvalet access modules to CA Endevor SCM CA Panvalet Interface modules:

1. Edit member BC1JPAN (in the JCL library) to include the name of your CA Panvalet load library.
2. Link-edit the PAM module, using BC1JPAN. Place the output load module that was created by this job into CSIQLOAD.
3. Redefine the Defaults Table, if necessary, to specify PV in the LIBENV parameter.

Edit BC1JPAN

Tailor the BC1JPAN JCL to provide the correct name of your CA Panvalet load library on the PAMLIB DD statement. The JCL for BC1JPAN is supplied on the installation tape.

Run BC1JPAN

If you have not already done so, copy your JOBCARD member to the beginning of BC1JPAN. Then submit the job for execution. BC1JPAN link-edits the PAM module and places the output load module in CSIQLOAD.

Set the CA Panvalet Parameters in the Defaults Table

Redefine the CA Endevor SCM Defaults Table, if necessary, to specify the value PV in the LIBENV parameter (in the TYPE=MAIN macro). Update the LIBENVP parameter to specify that CA Panvalet is installed at your site, as follows.

```
Col 1        Col 16                                  Col 72

             CIDEFLTS TYPE=MAIN,                     X
             .
             .
             .
             LIBENV=PV,                              X
             LIBENVP=Y,                              X
             .
             .
             .
```

Note: For more information about editing the table, see The Defaults Table (see page 207).

The ++CONTROL Password Interface

CA Endevor SCM provides a ++CONTROL password feature, which is available to the CA Endevor SCM for CA Panvalet Interface. This feature is optional, and allows CA Endevor SCM to supply the ++CONTROL password for each CA Panvalet library that will be accessed by CA Endevor SCM. The password interface is intended for CA Panvalet libraries that are currently password-protected.

The CA Librarian Interface

If your site is using CA Librarian as its source library manager, you need to install the CA Librarian Interface. This interface allows you to access members stored within the CA Librarian data sets.

CA Endevor SCM can add, update, or read from a CA Librarian data set. Whenever CA Endevor SCM adds or updates, it always adds or replaces the entire member, ignoring-but preserving- CA Librarian sequence numbers.

External and Internal Libraries

The CA Librarian data sets can be used as external libraries; for example, as development libraries, INCLUDE libraries, or target libraries.

It is *not* recommended that you use CA Librarian data sets as internal (base and delta) libraries, for performance and external usability issues. If you do use these data sets as CA Endevor SCM base and delta libraries, note that members written into the libraries are always stored with sequence numbers in columns 81-86, regardless of the type of language used. This is done to protect CA Endevor SCM information.

Set the CA Librarian Parameters in the Defaults Table

Redefine the CA Endevor SCM Defaults Table as follows, if necessary, to accommodate for the CA Librarian Interface:

- Specify the value LB in the LIBENV parameter (in the TYPE=MAIN macro).
- Update the LIBENVP parameter to specify that CA Librarian is installed at your site.
- Specify the name of the CA Librarian load module for your site in the LIBRPGM parameter. In the following example, the load module name is AFOLIBR, the default name delivered by CA Librarian .

```
Col 1         Col 16                                  Col 72

              CIDEFLTS TYPE=MAIN,                     X
              .
              .
              .
              LIBENV=LB,                              X
              LIBENVP=Y,                              X
              LIBPRGM=AFOLIBR,                        X
              .
              .
              .
```

Note: For more information about editing the table, see The Defaults Table (see page 207).

When to Customize for CA Librarian

Using CA Librarian may require some customization at your site. Because CA Endevor SCM automatically specifies columns 81-86 as the sequence columns for CA Librarian , and preserves all 80 columns of data (as applicable), it is not absolutely necessary to customize the interface at installation.

- If you do not require the Sequence Update facility (SEQUPD) of CA Librarian , you do not need to customize CA Endevor SCM to run the CA Endevor SCM CA Librarian Interface.
- If you need to control the sequence column numbers by type (if you are using the Sequence Update facility), you must customize your installation. Customization allows you to keep the CA Librarian members in external CA Librarian data sets, which may be updated by using imbedded sequence numbers.

For example, some sites have families of programs that are all updated by a common set of changes to a central copy of a particular type of program, such as a payroll program with several variants. The same set of changes can be applied to all programs by using the CA Librarian SEQUPD facility. In this example, customization would be required, as the client would need to retain control of the sequence number columns.

Considerations When Customizing for CA Librarian

If you decide to customize the CA Endevor SCM CA Librarian Interface for your site, you must consider the following:

- CA Endevor SCM must know the sequence columns for the element type being processed.
- CA Endevor SCM must preserve the original set of sequence numbers within the element, so subsequent sequence number updates will match correctly.
- CA Endevor SCM must always set the sequence columns when writing the element to an external CA Librarian data set, so CA Librarian will allow you to use these sequence columns for SEQUPD processing. This procedure informs CA Librarian that these specific columns are available, which in turn prevents automatic renumbering by CA Librarian.
- You must ensure that any element types customized for sequence number processing have valid, ascending numeric sequence numbers within the element.

How to Customize Your Site for CA Librarian

There are two steps involved in customizing your installation for the CA Endevor SCM CA Librarian Interface. These steps are described next:

1. Assemble and link-edit a language definition table to define the sequence columns by internal language for your installation.
2. Define distinct CA Endevor SCM types for those elements that will use SEQUPD processing.

Edit the C1LIBRSQ Table Member

You need to edit C1LIBRSQ to meet the requirements of your site.

After you complete the edits, use an SMP/E USERMOD to assemble and link-edit C1LIBRSQ or use the sample JCL BC1JTABL to assemble and link source module C1LIBRSQ outside of SMP/E. BC1JTABL is supplied in the installation library iprfx.iqual.CSIQJCL.

Edit these lines as follows:

- SYSLMOD DD-Replace iprfx.iqual.CSIQLOAD with the CSIQLOAD data set name you are using.
- C1LIBRSQ lines-Each C1LIBRSQ line defines the sequence number columns to be used for a particular internal language type. Enter the information required by your organization. Each line must specify a language (three characters in length, indicated by xxx), as well as two column numbers-a beginning column number (n1) and an ending column number (n2). You must enter one line per language. You can enter as many lines as necessary.
- The language types are matched against the PV/LB LANG field on the Type Definition panel.

The following list reflects the language codes CA Endevor SCM uses:

ASM (Assembler)

GOF

COB (COBOL)

JCL

DAT (Data)

PLF

FOR (FORTRAN)

PLI

FRG (FORTRAN G)

RPG

FRH (FORTRAN H)

TXT

GIS

VSB

The following examples illustrate acceptable C1LIBRSQ lines:

```
C1LIBRSQ LANG=COB,SEQCOLS=(1,6)
```

In this example, the language type is COBOL and the sequence number columns are 1 through 6.

```
C1LIBRSQ LANG=ASM,SEQCOLS=(73,80)
```

In this example, the language type is Assembler and the sequence number columns are 73 through 80.

Any languages not specified in the JCL are stored with sequence columns 81 through 86.

Note: If your site uses language codes other than those listed above, contact Technical Support at http://ca.com/support. An optional patch is available that allows you to modify the language codes in the JCL to accommodate the codes you are using.

Define CA Endevor SCM Types

You must determine which CA Endevor SCM types require the CA Librarian SEQUPD capability when you define element types for a system. These types require specific information to be entered on the Type Definition panel, as follows:

- The PV/LB LANG field must match one of the languages defined in the language definition table.
- The COMPARE FROM and COMPARE TO fields must include the sequence column fields defined for that language.
- Specify N for EXPAND INCLUDES, if the source output library is a CA Librarian data set. (Expanding INCLUDES makes the sequence numbers invalid.)

Note: If the language does not match or the sequence columns are not included within the compare columns (or both situations occur), CA Endevor SCM stores the sequence numbers in columns 81 through 86 when writing to external CA Librarian data sets.

Considerations When Defining Types

When defining types for the CA Endevor SCM CA Librarian Interface, keep in mind the following points.

For element types that are customized for the CA Endevor SCM CA Librarian Interface:

- Avoid EXPAND INCLUDES during add or retrieve processing where the target is a CA Librarian data set. Expanding INCLUDES makes the sequence numbers invalid.
- Ensure that members have valid sequence numbers in the sequence columns. If the numbers are invalid, CA Librarian rejects members during CA Endevor SCM retrieve processing.
- If a CA Endevor SCM member has invalid sequence numbers, retrieve that member into an IBM partitioned data set, adjust the sequence numbers, and add the member back into CA Endevor SCM.
- If using reverse formatted CA Librarian libraries, you should be aware of how -INC is handled. CA Endevor SCM will convert the '-' to a hex '02' and be added to the library; CA Endevor SCM does not want expansion to happen in the base library. Any CA Endevor SCM action will convert the hex '02' back to a '-'.

For element types that are not customized for CA Librarian:

- Set the CA Endevor SCM COMPARE columns to exclude CA Librarian sequence columns, to avoid sequence number edit rejections by CA Librarian.

Alternate ID Support

The alternate ID support feature allows a site to protect the CA Endevor SCM data sets (that is, base, delta, listing, and source output libraries, as well as Master Control Files, the package data set, and processor output libraries) from direct update by users.

When using alternate ID support, a separate user ID is defined to CA Endevor SCM. This separate or alternate ID is granted update authority for the CA Endevor SCM libraries. When CA Endevor SCM needs to update these libraries, the system does so using the authorization assigned to the alternate ID.

By defining an alternate ID and assigning it specific update authorization, you can prevent a user from inadvertently updating particular libraries.

Limited alternate ID support for UNIX files and directories can be enabled by turning on the ENABLE_ALTID_USS_SECURITY option in the Options table. For more information, see Data Set Security in the *Security Guide*.

Site-Defined Symbolics

Site-defined symbolics are user-defined symbolic values that you reference within dataset name specifications for base, delta, source output, include libraries, and processors (that is, you can use them wherever you can use CA Endevor SCM symbolics). At execution time, any site-defined symbolics referenced by a processor are stored with the processor symbolics in the component data. If a site-level symbolic is also specified as a processor symbolic, the processor symbolic (and processor symbolic override) take precedence.

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

Chapter 7: Using Site Symbolics

This section contains the following topics:

Site-Defined Symbolics

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

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

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

To implement site symbolics, you must define the symbolic and its data value in a table that is assembled and linked into an authorized load library. Once this is done, you must update the SYMBOLTBL parameter in the C1DEFLTS table with the name of the site symbolics table. These actions are described next.

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

Considerations When Defining Site Symbolics

Before defining the site symbolics, consider the following to ensure your success:

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

  ```
  &#VENDORLB
  ```

- SYMDATA values must be enclosed in quotes if the value contains ampersands or quotes. For example:

  ```
  the value '&&C1SU'. sets a value of &C1SU
  or
  the value 'CICS(''SP''),NODBCS' sets a value of CICS('SP'),NODBCS
  ```

- Quotes or ampersands imbedded in the SYMDATA value must be doubled. For example:

  ```
  the value '&&C1SU'. sets a value of &C1SU
  or
  the value '&&&&C1SU'. sets a value of &&C1SU
  or
  the value 'CICS(''SP''),NODBCS' sets a value of CICS('SP'),NODBCS
  ```

- If you need to continue a symbol definition on a second line, place a non-blank character in column 72 and begin the next line in column 16.

Define the Site Symbolics

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

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

The following information describes the preceding format:

symbolname

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

symbolvalue

The data value associated with the site symbolic is 1 to 70 characters in length, with no restrictions on the content of the data.

If you do not specify a data value for a symbolic, CA Endevor SCM treats it as a null variable. For example, the following symbolic contains a null value:

```
ESYMBOL SYMNAME=  APA,SYMDATA=' ',
```

For example:

```
$ESYMBOL SYMNAME=#VENDORLB,SYMDATA=SYS2.VENDOR.LOADLIB
```

Use JCL member BC1JTABL in *iprfx.iqual.CSIQJCL* to assemble and link-edit the site defined symbolics table. An alternative is to employ an SMP/E USERMOD to accomplish this.

Update C1DEFLTS

After creating the symbolics table, update C1DEFLTS to reflect the table name. Use the SYMBOLTBL= parameter to define the table name. The Site Information from the C1DEFLTS panel displays the parameter value in the SYMBOLICS Table field.

Chapter 8: Using CCIDs

This section contains the following topics:

CCIDs

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

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

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

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

CCIDs and Comment Fields

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

Last Action CCID and/or comment

The CCID and/or comment used the last time any action was performed that updated CA Endevor SCM in any way (every action except RETRIEVE). This CCID and/or comment is stored in the Master Control File.

Current Source CCID and/or comment

The CCID and/or comment used the last time a CA Endevor SCM action changed the source. This CCID and/or comment is stored in the Master Control File.

Generate CCID and/or comment

The CCID and/or comment used the last time a CA Endevor SCM action caused output processing to occur. Output processing occurs when a processor is run or an output data set is updated. This CCID and/or comment is stored in the Master Control File.

Retrieve CCID and/or comment

These fields only reflect the CCID and/or comment used during the RETRIEVE action if the last action at that stage was RETRIEVE. This CCID and comment are stored in the Master Control File. Viewing this information allows you to determine which elements have been retrieved for update by a particular project.

Source Delta CCID and/or comment

Each delta level contains the CCID and/or comment specified in the action that created the delta level.

Component List Delta CCID and/or comment

Each component list delta level contains the CCID and/or comment specified in the action that created the delta level.

Specify that CCIDs are Required

During your CA Endevor SCM implementation you should decide whether to require CCID and/or comments. You can require the use of CCIDs and/or comments on a system-by-system basis.

To specify that CCIDs and comments are required for a system

1. Select option **3** on the Environment Options Menu and press Enter.

   The System Request panel appears.

2. Type **U** in the COMMAND field, enter the environment name in the ENVIRONMENT field, and press Enter.

   The System Definition panel appears.

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

   Note: If you do not specify a CCID and/or comment when one is required for the action you are requesting, CA Endevor SCM displays the Action Prompt panel. To complete your action request, type a valid CCID and/or comment and press Enter.

When CA Endevor SCM Updates CCID Fields

You can specify a CCID and/or a comment when requesting CA Endevor SCM to perform the following actions:

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

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

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

The fields that CA Endevor SCM updates depend on the action you specify. The following sections on specific actions provide details on which of the CCID and/or COMMENT fields are updated, and under what circumstances.

Note: For more information about each action, see the *User Guide*.

Add Action CCID Updates

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

Specifying an Add Action for a New Element

When you specify a CCID and/or comment in an ADD action for a new element, CA Endevor SCM uses this CCID and/or comment to set the following fields:

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

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

Specifying an Add Action for an Existing Element

When you specify a CCID and/or comment in an ADD action for an existing element, CA Endevor SCM uses this CCID and/or comment to set the following fields:

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

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

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

Update Action CCID Updates

When you specify a CCID and/or comment in an UPDATE action for an existing element, CA Endevor SCM uses this CCID and/or comment to set the following fields:

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

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

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

Retrieve Action CCID Updates

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

Generate Action CCID Updates

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

Specifying a Generate Action without Copyback

When you specify a CCID and/or comment in a GENERATE action without copyback, CA Endevor SCM uses this CCID and/or comment to set the following fields:

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

Specifying a Generate Action with Copyback

When you specify a CCID and/or comment in a GENERATE action with copyback, CA Endevor SCM uses this CCID and/or comment to set the following fields:

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

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

Move Action CCID Updates

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

Specifying a Move Action without History

When you specify a CCID and/or comment in a MOVE action without history, CA Endevor SCM uses this CCID and/or comment to set the LAST ACTION CCID and/or COMMENT fields. CA Endevor SCM also performs the following actions:

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

Specifying a Move Action with History

When you specify a CCID and/or comment in a MOVE action with history, CA Endevor SCM uses this CCID and/or comment to set the LAST ACTION CCID and/or COMMENT fields. CA Endevor SCM also performs the following actions:

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

Transfer Action CCID Updates

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

Specifying a Transfer Action without History

When you specify a CCID and/or comment in a TRANSFER action without history, CA Endevor SCM uses this CCID and/or comment to set the following fields:

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

CA Endevor SCM also:

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

Specifying a Transfer Action with History

When you specify a CCID and/or comment in a TRANSFER action with history, CA Endevor SCM uses this CCID and/or comment to set the following fields:

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

CA Endevor SCM also performs the following actions:

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

When you specify SYNCHRONIZE along with the TRANSFER WITH HISTORY option, CA Endevor SCM creates a synchronize source delta level. When CA Endevor SCM creates this synchronize delta level, it performs the following actions:

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

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

Delete Action CCID Updates

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

Restore Action CCID Updates

When you specify a CCID and/or comment in a RESTORE action for an existing element, CA Endevor SCM uses this CCID and/or comment to set the following fields:

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

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

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

Summary CCID Impact Chart

The following chart summarizes the impact of CA Endevor SCM actions on the six fields that each action may update with the CCID and/or comment information that you specify for the action:

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

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

Predefining CCIDs

Predefining CCIDs allows you to perform the following:

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

You can predefine CCIDs in one of the following ways:

- Define a CCID definition data set within CA Endevor SCM.
- Use a product such as IBM Tivoli Information Management for z/OS; CA Endevor SCM provides an interface for this product.
- Create your own file; CA Endevor SCM provides a user exit capability so you can define your own CCID definition file.

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

CCID Validation

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

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

If the system you are working with is defined with CCID required, then a valid CCID must be specified for all actions requiring a CCID. If the system you are working with is not defined with CCID required, then the CCID will not be validated and any value including no CCID (spaces) for the action is allowed.

CCID validation applies only to specific actions. This processing is not applicable with the LIST or PRINT actions, because you cannot specify a CCID for these actions. In addition, CCID validation is not performed for the MERGE action.

Note: CCID validation performed by CA Endevor SCM can be used as a secondary validation mechanism that is invoked if a no match condition is found in the CA Endevor SCM InfoMan Interface. For more information, see the *InfoMan Interface Administration Guide*.

CCID Definition Data Set

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

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

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

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

Create a CCID Definition Data Set

To create a CCID definition data set

1. Specify a CCID data set name in the Defaults Table.

   Note: When you create the sequential file, be sure to specify a data set name in the format: project.dataset.type, where *type* is a unique value. This causes the ISPF editor to use a unique profile for the data set.

2. Allocate that data set (using standard ISPF data set utilities).

3. Initialize the data set by copying member SAMPCIPO from the iprfx.iqual.CSIQJCL library into the allocated data set.

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

The Purpose of a Sequential Data Set

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

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

Note: To assist in the building of the application program, an assembler macro has been supplied with the system. The macro can be found in the installation source library (*iprfx.iqual.CSIQOPTN*) built during Step 1 of the CA Endevor SCM installation process. The member name of the macro is $CIPOREC.

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

Editing the File Using the ISPF Text Editor

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

First, enter the following primary commands:

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

Second, set your tab positions as follows:

```
*       *     *     * *     *     *     *

3       16    25    34 36   45    54    63
```

Using the CCID Definition Data Set

The CCID definition data set can be used for the following:

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

  If a CCID gets a "no match" condition from the CA Endevor SCM InfoMan interface, CA Endevor SCM uses the CCID definition data set to check further. If the interface recognizes the CCID, then CA Endevor SCM bypasses the CCID check.

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

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

- Environment = **DEMO**
- Stage number = **1**
- Systems for TAX-CHNG-89 = **PAYROLL** and **ACCOUNTG**
- Systems for NEW-COMPPLAN = **PAYROLL** and **PERSONEL**

The CCID definition file would be coded as:

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

Sample CCID Definition Data Set

The following is a sample CCID definition data set.

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

The following coding conventions apply to this data set:

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

A description of each field and its card position are shown in the data set that follows:

CCID (3)

The name of the CCID. The name must be fully specified, defining a valid CCID. This value must be specified on every definition line. The same CCID may appear on multiple lines.

Userid (16)

(Optional) If used, only the users whose user ID is specified (whether by full name or with the use of a name mask) may use the CCID indicated on this line.

Environ (25)

(Optional) Environment name. If used, only elements in the environment specified (whether by full name or with the use of a name mask) can be updated under the CCID indicated on this line.

 # (34)

(Optional) Stage number. If used, only elements in the stage specified (either 1, 2, or *) can be updated under the CCID indicated on this line.

System (36)

(Optional) System name. If used, only elements in the system specified (whether by full name or with the use of a name mask) can be updated under the CCID indicated on this line.

Subsys (45)

(Optional) Subsystem name. If used, only elements in the subsystem specified (whether by full name or with the use of a name mask) can be updated under the CCID indicated on this line.

Type (54)

(Optional) Element type. If used, only elements of the type specified (whether by full name or with the use of a name mask) can be updated under the CCID indicated on this line.

Element (63)

(Optional) Element name. If used, only elements whose names are specified (whether by full name or with the use of a name mask) can be updated under the CCID indicated on this line.

Chapter 9: Using SMF Recording

This section contains the following topics:

Enable SMF Recording

The SMF option records each action and each security violation that occurs during CA Endevor SCM processing. SMF recording is done on an Environment-by-Environment basis. Therefore, you can selectively enable SMF recording for each of your Environments. The SMF option can write out the following SMF records:

- Security records— For each security violation, or each error that is returned from the security exit (exit 1).
- Activity records— For each action executed, except for the following actions: Print, Display, Copy, and List. Activity records are written out at the end of action processing.

The administrator enables the SMF option in the Defaults table (C1DEFLTS). Per Environment, recording can be enabled for security records, activity records, or both.

Note: This topic assumes you are working with the native security facility. If the optional External Security Interface (CA Endevor SCM ESI) is installed at your site, see the *Security Guide* for related information.

Follow these steps:

1. Edit the SMFREC# parameter in the C1DEFLTS TYPE=MAIN macro to specify an SMF record number for for your site.

   SMFREC#— Specifies the SMF record number assigned to SMF records written by CA Endevor SCM at this site. Default is 0, which disables SMF recording.

   Talk to a systems programmer at your site to ensure that you are collecting the SMF records related to the record number specified in the SMFREC# parameter. Recording records is controlled in SYS1.PARMLIB(SMFPRMxx).

2. Edit the following parameters in the C1DEFLTS TYPE=ENVRNMNT.

   SMFACT— Specifies whether to write SMF activity records for this Environment.

   Y— Writes activity records for the Environment.

   N— Does not write activity records for the Environment. Default is N.

SMFSEC— Specifies whether to write SMF security records for this Environment.

Y—Writes security records for the Environment.

N— Does not write security records for the Environment. Default is N.

3. Reassemble and link C1DEFLTS to activate the modifications.

Note: You can use the CSV utility to extract SMF record data corresponding to activity related to CA Endevor SCM actions or to security violations recorded by CA Endevor SCM. For more information, see the chapter Using the Comma Separated Value (CSV) Utility in the *Utilities Guide*.

Record Formats

CA Endevor SCM uses several blocks to build the SMF security and activity records, each comprising one or more DSECTs. All SMF DSECTs are supplied by the following macros in iprfx.iqual.CSIQOPTN: $SMFBKDS, $SMFHDDS, $SMFREC1, $SMFREC2, $SMFREC4.

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

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

- DSECT $SMFREC1 for security records.
- DSECT $SMFREC2 for activity records. For activity records, the $SMFREC2 block encompasses one or more action-specific blocks.

Note: For more information about each DSECT used to format SMF records, see DSECT Descriptions (see page 176).

SMF Security Records

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

- $SMFHDDS
- $SMFREC1

SMF Activity Records

SMF activity records include data specific to the action being reported, and applies to all actions except PRINT, DISPLAY, COPY, and LIST. Each activity record has one occurrence of the $SMFREC2 DSECT, and one or more action-specific blocks from the $SMFBKDS macro, as appropriate to the action. Each action-specific block has an action-block header (DSECT SM2BHDDS), and either the type 1, 2, 3, 4 or 5 action-block detail information, respectively DSECT SM2ENVDS, SM2LCGDS, SM2LPRDS, SM2REQDS or SM2ALTDS.

Note: For more information about each action block, identified separately, see $SMFBKDS DSECT: Action-Specific Blocks (see page 182).

The combined format for SMF activity records, is described using the following DSECTs:

- $SMFHDDS
- $SMFREC2
- SM2BHDDS action-block header
- $SMFBKDS action-block detail (type n, includes up to five action blocks)

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

Action Block Detail Type #	DSECT Type	Actions	Description
1	Environment (SM2ENVDS)	Add (2) Alter Update (2) Retrieve (2) Generate (2) Move (2) Delete (2) Transfer (2) Archive (2) Restore (2)	Information describing the Environment where the action took place: name of the Environment, site ID, and so on. Where two blocks are used, one represents the source Environment for the Element, while the other represents the target Environment.

Action Block Detail Type #	DSECT Type	Actions	Description
2	Last Change (SM2LCGDS)	Add Update Generate Move Delete Transfer Restore	Information about the last processor executed: processor name, date and time of execution, etc. Used for actions that can invoke a processor.
3	Processor Info (SM2LPRDS)	Add Update Generate Move Delete Transfer Restore	Information about the last processor executed: processor name, date and time of execution, etc. Used for actions that can invoke a processor.
4	Request Parameter Info (SM2REQDS)	All actions	General information about the action: CCID, comments, and so forth.
5	Alter Info (SM2ALTDS)	Alter	Information specific to the Alter action: Before field value, after field value, and match (mask) value.

DSECT Descriptions

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

SMF Header Block Field Descriptions

The header block, $SMFHDDS DSECT, is written at the beginning of each SMF record. CA Endevor SCM security and activity records can be identified by the record type field in the header, SMHRECTY, which has the same value in every CA Endevor SCM SMF record. This value is defined in the Defaults Table (defaults to 230).

The following information describes the SMF header block fields:

SMHLEN

Size of the (security or activity) record, in bytes. This includes the size of the header block as well as the data block:

- $SMFREC1-Security records
- $SMFREC2 (including all action-specific blocks)-activity records

SMHSEGID

Not used.

SMHSID

Operating system against which the SMF record is written. Always X'02' with release 3.7.

SMHRECTY

Number that identifies this SMF record as specific to CA Endevor SCM. This number is defined to CA Endevor SCM through the Defaults Table.

SMHTIME

Time the SMF record was created (binary time of day in 100ths of a second-1 = .01 second).

SMHDATE

Date the SMF record was created (packed yyddd).

SMHCPUID

Number to identify the CPU model on which CA Endevor SCM is running.

SMHHDLEN

Size of the header block (DSECT $SMFHDDS), in bytes.

SMHC1VER

Code that identifies the format of the SMF record.

- SMH$RF00-This record was created by CA Endevor SCM release 2.2 or a prior release.
- SMH$RF01-This record was created by CA Endevor SCM release 2.5 or a later release.

SMHCTLTY

Code that identifies the next record format in the SMF record.

SMHCONT

Not used.

SMHSEQ

Not used.

SMHC1ENV

Environment name associated with the current processing.

SMHUSER

User ID associated with the current processing.

SMHSIZE

Size of the $SMFHDDS DSECT.

Security Record Data Block Field Descriptions

The security record data block, $SMFREC1 DSECT, applies to security records and includes data specific to the security violation being reported.

The following information describes the fields in the security record data block:

SM1RECLN

Size of this data block (DSECT $SMFREC1), in bytes.

SM1RECVN

Version number. Identifies the CA Endevor SCM release that created the record. Valid values are:

- 1— The record was created by a release of CA Endevor SCM prior to release 4.0.
- 2— The record was created by CA Endevor SCM release 4.0 or a later release.

SM1FUNC

Number that identifies the current activity: generally the type of action.

SM1FUNNM

Applicable if SM1FUNC references a CA Endevor SCM action. Name of the current action (ADD, UPDATE, etc.).

SM1ERRCD

The 4 characters that are used to construct the error code written to the CA Endevor SCM Execution Report for the current action in the ECBMSGCD field. The error code is formatted as C1XNNNNS, where:

- X-Describes the origin of the message.
- NNNN-Defined by this field.
- S-Severity code associated with the return code field, ECBRTCD, in DSECT $ECBDS.

SM1ERRLN

Not used.

SM1ERMSG

Error message associated with the current activity for the ECBMSG field.

SM1SITE

CA Endevor SCM site ID, as defined to the Defaults Table.

SM1STGID

Stage number for the current action request: 1 or 2.

SM1STGCD

Applicable if SM1FUNC references a CA Endevor SCM action. Stage ID for the current action request.

SM1IFUNC

Action information that is used internally by CA Endevor SCM.

SM1ENVM

Environment name for the current processing.

SM1STAGE

Applicable if SM1FUNC references a CA Endevor SCM action. Stage name for the current action request.

SM1SYSTM

Applicable if SM1FUNC references a CA Endevor SCM action. System name for the current action.

SM1SUBSY

Applicable if SM1FUNC references a CA Endevor SCM action. Subsystem name for the current action.

SM1STYPE

Applicable if SM1FUNC references a CA Endevor SCM action. Element type for the current action.

SM1ELEMT

(SM1RECVN version 1 records only) Applicable if SM1FUNC references a CA Endevor SCM action. Name of the Element specified for the current action.

SM1EAOFF

(SM1RECVN version 2 records only) Element area offset. When the offset is added to the beginning of the record's address, the resulting address points to an area with in the SM1BFAREA. The Element area is comprised of two adjacent components:

- Two byte field containing the length
- Element name

SM1DSNAM

(SM1RECVN version 1 records only) Applicable for security violations that relate to ADD, UPDATE, RESTORE, ARCHIVE, and RETRIEVE action requests. Data set name associated with the external file used by the current action.

SM1FAOFF

(SM1RECVN version 2 records only) File area offset. When the offset is added to the beginning of the record's address, the resulting address points to an area with in the SM1BFAREA. The file area is comprised of two adjacent components:

- Two byte field containing the length
- File name

The file name may be a traditional file or an USS path name.

SM1DSMEM

(SM1RECVN version 1 records only) Applicable for security violations that relate to ADD, UPDATE, RESTORE, ARCHIVE, and RETRIEVE action requests. Member name associated with the Element for which the current action applies, within the above data set. Applicable when the data set is a library.

SM1NAOFF

(SM1RECVN version 2 records only) Name area offset. When the offset is added to the beginning of the record's address, the resulting address points to an area with in the SM1BFAREA. The name area consists of two adjacent components:

- Two byte field containing the length
- File name

The name may be a member name or an USS file name.

SM1BSIZ

Base size of the record.

SM1BFAREA

(SM1RECVN version 2 records only) This is reserved record space containing the various areas such as Element, file and name.

SM1SIZ

Size of the $SMFREC1 DSECT

Activity Record Data Block Field Descriptions

The activity record data block, $SMFREC2 DSECT, applies to activity records and includes information specific to the action being reported. Depending on the action type, one or more action-specific blocks are incorporated at the end of this DSECT.

The following information describes the activity record data block fields:

SM2RECLN

Size of this data block (DSECT $SMFREC2), in bytes, including all action-specific blocks included at the end (DSECT $SMFBKDS).

SM2RECVN

Version number to identify this data block ($SMFREC2). This value was updated to 2 for addition of the Alter block. Use the SM2$CURV label to detect that either the record is the current version (2) or the previous version of the SMF activity record.

SM2FUNC

Number that identifies the current activity: generally the current type of action.

SM2FUNNM

Applicable if SM2FUNC references a CA Endevor SCM action. Name of the current action (ADD, UPDATE, and so forth).

SM2BLKS

Count of action-specific blocks included at the end of this DSECT: 1-5.

SM2RECHD

Size of this data block (DSECT $SMFREC2), in bytes, excluding the action-specific blocks (DSECT $SMFBKDS).

SM2IFUNC

Action information that is used internally by CA Endevor SCM.

SM2HDRLN

Size of the $SMFREC2 DSECT, excluding the action-specific blocks

SM2SIZ

Size of the $SMFREC2 DSECT, including all action-specific blocks incorporated at the end.

$SMFBKDS MACRO: Action-Specific Blocks

The action-specific block applies for activity records, and includes data specific to the action being reported. $SMFBKDS includes six DSECTs-one action-block header and five action-block detail DSECTs. These DSECTs are used in pairs, with each pair including a header and one type 1, 2, 3, 4 or 5 detail DSECT. The type of detailed information included can be identified through the SM2BTYP field in the header.

Action-Block Header Field Descriptions (SM2BHDDS)

The following information describes the fields in action-block header:

SM2BLEN

Size of this action-specific block (action-block header, SM2BHDDS, plus the action-block detail DSECT), in bytes.

SM2BTYP

Code that indicates the DSECT used to write out the action-block detail for this action- specific block:

SM2BVER

Version number to identify this DSECT ($SMFBKDS). This should always be 1 except for the action block header.

SM2BFLG1

Not used.

SM2BFLG2

Not used.

SM2BHDR

Size of the action-block header ($SM2BHDDS).

Environment Action-Block Detail Field Descriptions (SM2ENVDS)

The following information describes the Environment action-block detail fields:

SM2SITE

CA Endevor SCM site ID, as defined to the Defaults Table.

SM2STGCD

Stage ID for the current action request.

SM13STGID

Stage number for the current action request: 1 or 2.

SM2SRCHR

Search and Replace utility flag.

SM2$ELEM

Search and Replace utility action on an Element.

SM2$COMP

Search and Replace utility action on an output component.

SM2VER

Version number associated in the MCF with the Element for the action request.

SM2LVL

Number to identify the current level of the Element for the current action request.

SM2STAGE

Stage name for the current action request.

SM2SYSTM

System name for the current action request.

SM2SUBSY

Subsystem name for the current action request.

SM2STYPE

Element type for the current action request.

SM2ELEMT

(Version 1 Record Only) Name of the Element specified for the current action request.

SM2EAOFF

(Version 2 Record Only) Element area offset. The Element name for the current action request is stored in the buffer area located near the end of the record. When the offset is added to the beginning of this record's address, the resulting address points to an area within SM2EAREA buffer space. The Element area is composed of two adjacent components:

- Two-byte length field
- Element name

SM2PRGRP

Processor group name for the current action request.

SM2DSNAM

(Version 1 Record Only) Applicable for ADD, UPDATE, RESTORE, ARCHIVE, and RETRIEVE action requests. Data set name associated with the external file used by the current action.

SM2FAOFF

(Version 2 Record Only) File area offset. The file name for the current action request is stored in the buffer area located near the end of the record. When the offset is added to the beginning of this record's address, the resulting address points to an area within SM2EAREA buffer space. The file area is composed of two adjacent components:

- Two-byte length field
- Data set name

The file name may be a traditional data set or an USS path name.

SM2DSMEM

(Version 1 Record Only) Applicable for ADD, UPDATE, RESTORE, ARCHIVE, and RETRIEVE action requests. Member name associated with the Element for which the current action applies, within the above data set. Applicable when the data set is a library; blank otherwise.

SM2NAOFF

(Version 2 Record Only) Name area offset. The member name for the current action request is stored in the buffer area located near the end of the record. When the offset is added to the beginning of this record's address, the resulting address points to an area within SM2EAREA buffer space. The name area is composed of two adjacent components:

- Two-byte length field
- Member name (PDS file) or file name (HFS)

If a member/file name is not applicable to the current action, the offset value is set to zero.

SM2EAREA

(Version 2 Record Only) Space reserved in the record containing various data such as the Element, file and member name.

SM2ENVLN

Size of the action-specific block (header and detail) if the action-block detail uses the Environment DSECT ($SM2ENVDS).

Last Change Action-Block Detail Field Descriptions (SM2LCGDS)

The following describes the Last Change action-block detail fields:

SM2CCOM

Current-level comment for the Element.

SM2MLDTE

Current-level date for the Element.

SM2MLTME

Current-level time for the Element.

SM2LUSID

Current-level user ID for the Element.

SM2MLACT

Last action executed for the Element.

SM2MLTOT

Count of source statements for the Element, as of the current level.

SM2LCHLN

Size of the action-specific block (header and detail) if the action-block detail uses the Last Change DSECT ($SM2LCGDS).

Processor Information Action-Block Detail Field Descriptions (SM2LPRDS)

The following describes the Processor Information action-block detail fields:

SM2LPRON

(Element) name of the processor last run for the Element.

SM2LPROD

Processor date for the Element.

SM2LPROT

Processor time for the Element.

SM2LPROU

Processor user ID for the Element.

SM2LPHRC

Processor return code for the Element.

SM2LPRC1

CA Endevor SCM return code for the Element.

SM2PRCLN

Size of the action-specific block (header and detail) if the action-block detail uses the Processor Information DSECT ($SM2LPRDS).

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

The following describes the Request Parameter Info action-block detail fields:

SM2RCCID

CCID associated with the current action request.

SM2RCOMM

Comments associated with the current action request.

SM2RFLAG

Not used.

SM2RSISO

Indicates whether the current action requested a signout override: Y (Yes) or N (No).

SM2RSICO

Applicable for a RETRIEVE action. Indicates whether the current action specified copy-only: Y (Yes) or N (No).

SM2REXPN

Applicable for a RETRIEVE action. Indicates whether the current action requested that INCLUDEs be expanded: Y (Yes) or N (No).

SM2ROVER

Applicable for a RETRIEVE action. Indicates whether the current action requested an overwrite if a member by the same name exists already in the output library: Y (Yes) or N (No).

SM2RRTCD

The return code associated with the current action request.

SM2RDEL

Indicates whether the Element was deleted upon completion of the requested action: Y (Yes) or N (No). This value is X'40' if a delete does not apply for the current action.

SM2REQLN

Size of the action-specific block (header and detail) if the action-block detail uses the Request Parameter Information DSECT ($SM2REQDS).

SM2REMCH

Indicates whether the Validate action with the Element Master option was specified. Possible values are: Y (Yes) or N (No).

Note: This field is currently not in use and will not be populated.

SM2RSYCH

Indicates whether the Validate action with the Synchronization option was specified. Possible values are: Y (Yes) or N (No).

Note: This field is currently not in use and will not be populated.

SM2RCMCH

Indicates whether the Validate action with the Component Validation option was specified. Possible values are: Y (Yes) or N (No).

Note: This field is currently not in use and will not be populated.

SM2RTMSG

Indicates whether the Validate action with the Terse option was specified. Possible values are: Y (Yes) or N (No).

Note: This field is currently not in use and will not be populated.

SM2RAUTG

Indicates whether the AUTOGEN Element option was specified on the ADD, UPDATE or GENERATE action. Possible values are: Y (Yes) or N (No). If the value of this field is Y, the value of the SM2AUTX field will be N.

SM2RAUTX

Indicates whether this where-used Element was generated on behalf of a component Element. Possible values are: Y (Yes) or N (No). If the value of this field is Y, the value of the SM2AUTG field will be N.

SM2RNSRC

Indicates whether the NOSOURCE action option was in effect for the Generate, Add, or Update action. Possible values are: Y (Yes) or N (No).

SM2RSPAN

Indicates whether one of the Autogen Span options was specified on the Add, Update or Generate action. Possible values are: N (span none), A (span all), S (span Systems), B (span Subsystems).

Alter Action Information Block Field Descriptions (SM2ALTDS)

The following information describes the Alter action-block detail fields:

SM2AFLD

Contains the field ID requested for an Alter action. An SMF record is created for each field requested in an Alter action, meaning that one Alter action can create several Alter SMF records.

Note: An SMF record is only created if the Alter action is performed and the Master Control File is updated. No SMF record is created in the following cirsumstances:

- For the Noupdate mode
- If none of the requested fields matched current field values, so that no Master Control File fields were updated
- If a prior error occurred that cancelled the action and no Master Control File fields were updated

Possible values are as follows:

1— Indicates the Alter action requested the replacement of a Generate CCID value in the Master Control File.

2— Indicates the Alter action requested the replacement of a Last Action CCID value in the Master Control File.

3— Indicates the Alter action requested the replacement of a Retrieve CCID value in the Master Control File.

4— Indicates the Alter action requested the replacement of a Description value in the Master Control File.

5— Indicates the Alter action requested the replacement of a Processor Group value in the Master Control File.

6— Indicates the Alter action requested the replacement of a Signout Userid value in the Master Control File.

7— Indicates the Alter action requested the replacement of a User Data value in the Master Control File.

SM2AUPDT

Indicates whether the Master Control File was updated. Valid values are Y or N:

Y— Indicates that the Master Control File record was updated.

N— Indicates that the Master Control File record was not updated.

SM2ADATA

Label for the beginning of the variable Alter data area. For the Alter action data area, only one group of fields exists per record, the data area is redefined for each group of fields. These fields correspond to the Alter action field IDs, refer to SMA2FLD for field ID values.

Generate CCID fields:

SM2ACCGB

Value of the Generate CCID before the Alter action.

SM2ACCGA

Value of the Generate CCID after the Alter action.

SM2ACCGM

Mask value specified as the from-value for the Generate CCID.

Last Action CCID fields:

SM2ACCLB

Value of the Last Action CCID before the Alter action.

SM2ACCLA

Value of the Last Action CCID after the Alter action.

SM2ACCLM

Mask value specified as the from-value for the Last Action CCID.

Retrieve CCID fields:

SM2ACCRB

Value of the Retrieve CCID before the Alter action.

SM2ACCRA

Value of the Retrieve CCID after the Alter action.

SM2ACCRM

Mask value specified as the from-value for the Retrieve CCID.

Description fields:

SM2ADESB

Value of the Description before the Alter action.

SM2ADESA

Value of the Description after the Alter action.

SM2ADESM

Mask value specified as the from-value for the Description.

Processor Group fields:

SM2APRGB

Value of the Processor Group before the Alter action.

SM2APRGA

Value of the Processor Group after the Alter action.

SM2APRGM

Mask value specified as the from-value for the Processor Group.

Signout Userid fields:

SM2ASGNB

Value of the Signout Userid before the Alter action.

SM2ASGNA

Value of the Signout Userid after the Alter action.

SM2ASGNM

Mask value specified as the from-value for the Signout Userid.

User Data fields:

SM2AUSRB

Value of the User Data before the Alter action.

SM2AUSRA

Value of the User Data after the Alter action.

SM2AUSRM

Mask value specified as the from-value for the User Data.

SM2AUSR_FPOS

From-position specified for the User Data.

SM2AUSR_FTXL

Length of the mask value (from-value) text string for the User Data.

SM2AUSR_TPOS

To-position specified for the User Data.

SM2AUSR_TLEN

To length specified for the User Data.

SM2AUSR_TTXL

Text length of the replacement value specified for the User Data.

SM2AUSR_TPAD

Pad character specified for the User Data. This field is required if the “to” length (SM2AUSR_TLEN) is greater than the length of the replacement value (SM2AUSR_TTXL)

Chapter 10: Performance Improving Options

Note: The *Scenario Guide* contains information that previously appeared in this chapter. You can find the following features and their corresponding scenario-based knowledge documents in the *Scenario Guide*.

- Global Type Sequencing— *How to Enable Global Type Sequencing*
- Concurrent Action Processing— *How to Enable Concurrent Action Processing*
- Autogen action option— *How to Automatically Generate Using Elements*

This section contains the following topics:

Virtual Inventory Configuration NoSource Option

The *Virtual Inventory Configuration* feature enables unchanged source code to be compiled and linked at a CA Endevor SCM inventory location without the requirement that the source code be copied back to that inventory area. This feature is triggered by the NoSource option on the Generate action, provided the element does not exist at the target location. NoSource can be specified on foreground or batch Generate actions. The C1BASELIB symbolic, which references the base type library, can still be used in your processor. The CONWRITE processor utility has been modified to solve the current source for the sourceless elements.

The elimination of the requirement to fetch back unchanged source code dramatically improves concurrent development productivity by reducing the possibility of out-of-sync changes being made to different copies of the same source code. For example, with this option, source code can be fetched back to a sandbox work area only when source changes are required. Another benefit is improved performance resulting from the elimination of the movement (fetch and promotion) of unchanged source code. In addition, virtual inventory configuration is used by the new Autogen action option to facilitate processing of Generate actions.

When NoSource is used, the target location coded in a Generate action contains the outputs created by the Generate processor. In addition, a *sourceless element* is created at the target. Because the element source is not fetched back to the target location, the MCF record for the element generated at the target identifies it as a sourceless element. The MCF element contains the last level timestamp of the upstream element. Base and deltas do not exist for sourceless elements, so these fields are blank in the MCF. When actions are performed against sourceless elements, the source from the next sourced element upstream from it will be used providing that the last level timestamps are equal.

How Virtual Inventory Configuration Works

For sourceless elements, CA Endevor SCM will use the next sourced element from up the map as the source for the generate processor. This is possible because CA Endevor SCM generate processors, for reverse and image delta elements, use the actual source type definition‘s base library data set name as the location of the element‘s source. This library data set name is associated with the CA Endevor SCM symbol known as C1BASELIB. Another way of extracting the source for an element in a processor, whether it is sourced or not, is to use the Conwrite processor utility. Conwrite is able to resolve the source for sourceless elements by looking upstream for the next sourced element.

Assuming that the element source is not at the target location and the element source is located up the map from the target, the effect of the Generate NoSource action is as follows:

1. The element's source is not fetched back to the target location from up the map.
2. The first occurrence of the element up the map from the target location is used as input to the generate processor. This element source is used as follows:
    a. All target inventory C1 symbols are set with the element's source inventory fields before doing the C1BASELIB substitution.
    b. The C1BASELIB symbol is set with the library base data set defined at the source location (up the map).
    c. After the C1BASELIB symbol is resolved, the target inventory C1 symbol is reset to the action's target inventory.
3. After the Generate action completes, the targeted location coded in a Generate action contains the outputs created by the generate processor. The MCF element created at the target location contains data similar to a fetched back element except that the element base and delta name fields are blank and the record is marked as a sourceless element.

Note: Elements with forward deltas must use the CONWRITE utility in their processors to rebuild the element source. This utility supports sourceless elements by using the actual source location from up the map to rebuild the element. Conwrite also supports forward and reverse deltas.

Chapter 11: Using the User Options Menu Facility

This section contains the following topics:

Menu Functions

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

- Option 1-Build and submit CA Endevor SCM batch report requests
- Option 2-Invoke the ACM Query Facility

The User Options Menu is delivered as member NDVRUSER in the iprfx.iqual.CSIQPENU library, and this library also contains the source. The CLISTs, panels, messages, and skeleton JCL for the REPORTS option on this menu are delivered in the following libraries:

- CLISTs are stored in the iprfx.iqual.CSIQCLS0 library as members C1SR1000; C1SR1100; C1SR1200; C1SR1300; C1SR1400; and C1SR1500.
- Panels are stored in the iprfx.iqual.CSIQPENU library as members C1SR1000; C1SR1100; C1SR1200; C1SR1300; C1SR1400; and C1SR1500.
- Messages are stored in the iprfx.iqual.CSIQMENU library as member CISR00.
- Skeleton JCL is stored in the iprfx.iqual.CSIQSENU library as members C1SR8000 and C1SR8100.

The User Option Menu Panel

The User Option Menu panel is delivered with functionality to invoke a report CLIST, the ACM Query CLIST. You may decide to modify this panel to remove selections not installed at your site or to add additional CA Endevor SCM related selections for features you develop.

If you decide to customize the User Options Menu, we recommend you do so within CA Endevor SCM. This ensures any changes you make are recorded as change levels.

Add an Option to the User Options Menu

To add an option on this panel

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

   ```
   % U+NDVRUTL - CA Endevor SCM Client Utility
   ```

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

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

Remove an Option from the User Options Menu

To remove options on this panel

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

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

ISPF panels support a maximum of 24 display lines. The delivered panel contains 24 lines. Line 23 contains the text "Enter END to return" and the 24th line is left blank to support split screen mode. It is strongly recommended that you do not alter the number of lines in the)BODY section. If the panel contains more than 24 lines, ISPF issues an error message when attempting to display the panel. If it contains less than 24 lines, ISPF adds blank lines to the end of the panel. Avoid these errors by:

- Replacing a deleted line with a blank line. The line must contain the % attribute character in column 1.
- Deleting a blank line when adding a new line.

Chapter 12: Using the ISPF Dialog Options Configuration Table

This section contains the following topics:

Configuring ISPF Dialog Options

This feature allows you to establish the default values that appear automatically on the foreground action panels. During a CA Endevor SCM session, however, a user can still change the value for one or more options on an individual panel(s).

The dialog options configuration table feature allows you to override default values set for specific options associated with CA Endevor SCM actions. The defaults are assigned in an ISPF dialog configuration table that is shipped with the CA Endevor SCM installation files. You can override the assigned default values by updating and assembling the ISPF configuration table, ENDICNFG.

If you do not want to change the dialog default field values, you do not need to take any action. CA Endevor SCM uses the configuration table that is shipped with the installation tape.

Note: You can set default values for all Y/N action option fields and for most list fields. You can also set the default package selection fields. For a list of the dialog fields that can be updated, see Dialog Options Fields. In addition, you cannot set default values for the CCID, COMMENT, PROCESSOR GROUP fields, the list option WHERE fields, or data set name fields.

The Default Configuration Table

The CA Endevor SCM files contain the default configuration table, ENDICNFG, and the JCL you can use to reassemble and link-edit the table. This table contains the existing default values.

To change the default of any dialog field, proceed as follows:

1. Edit the parameters you want to change.
2. Reassemble and link-edit the configuration table.

Note: The default configuration table is regularly updated until the time of shipping. For the most recent version of ENDICNFG, see the member ENDICNFG in iprfx.iqual.CSIQSRC.

Dialog Options Fields

Each option is a dialog field in the configuration table. There are two kinds of fields: those that affect how CA Endevor SCM processes the action and those that affect the information returned or presented by CA Endevor SCM.

The following table lists each of the CA Endevor SCM dialog fields for which your site can override the default value. The first column contains the field names as they appear in the configuration table. These names map to the action option or list option names that appear on the element or package action panels. The last column in the table indicates whether this is an action option (A), a list or filter option (F), or an option that influences CA Endevor SCM's behavior (O).

ISPF Dialog Field Name	Valid Values	Current Default	Panel Field Description	Action, Filter, or Other
ACKNOWLEDGE_ ELEMENT_JUMP	Y or N	N	Acknowledge element jump.	A
APPEND_SCL	Y or N	N	Default value for the “APPEND” field on the Batch Options panel.	F
APPEND_SCL_PKG	Y or N	N	Default value for the “APPEND TO PACKAGE” field on the Create/Modify Package panel.	F
APPROVED	Y or N	Y	Include APPROVED packages.	F

ISPF Dialog Field Name	Valid Values	Current Default	Panel Field Description	Action, Filter, or Other
AUTOGEN	Y or N	N	When acting on an element, automatically generate the elements that use this component element. Option for ADD, UPDATE and GENERATE actions only.	A
AUTOGEN_SPAN	NONE, ALL, SYS, SUB	NONE	To be used with the AUTOGEN option. Automatically generates using elements found in Systems and Subsystems outside the logical map of the AUTOGEN'd component; NONE for no SPAN option, ALL to span all Systems and Subsystems; SYS to span only Systems with the same Subsystem as the component; or SUB to span all Subsystems within the System of the component.	A
BROWSE_VIEW_MODE	B or V	B	Default display mode; B for browse, V for view.	O
BUILD_USING_MAP	Y or N	Y	Build the element selection list using the Environment Map.	F
COMMITTED	Y or N	Y	Include COMMITTED packages.	F
COMPONENTS_ONLY	Y or N	N	Delete only the element Component List.	A
COPYBACK	Y or N	Y	Copy an element from up the map during generate processing.	A
DELETE_AFTER_ ARCHIVE	Y or N	N	Delete the element or package after the ARCHIVE action completes.	A
DELETE_AFTER_ MOVE	Y or N	Y	Delete the source element after the MOVE action completes.	A
DELETE_AFTER_ TRANSFER	Y or N	Y	Delete the source element after a TRANSFER action completes.	A
DELETE_INPUT_ SOURCE	Y or N	N	Delete the input source after an ADD action completes.	A
DELETE_MODE	F or B	F	Default mode when executing the delete action in Quick-Edit; F for foreground, B for batch.	A

ISPF Dialog Field Name	Valid Values	Current Default	Panel Field Description	Action, Filter, or Other
DELETE_PROMHIST	Y or N	N	Delete all the promotion history associated with the package when the Commit action completes.	A
DENIED	Y or N	Y	Include denied packages	F
ENABLE_BACKOUT	Y or N	Y	Enable package backout	A
ENTERPRISE_PKG	A or E or X	A	ISPF Package Selection list filter option. If A, lists enterprise and non-enterprise packages. If E, lists enterprise packages only. If X, lists non-enterprise packages only.	F
EXECUTED	Y or N	Y	Include executed packages.	F
EXPAND_INCLUDES	Y or N	N	Expand imbedded INCLUDES during a RETRIEVE action.	A
GENERATE_ELEMENT	Y or N	Y	Generate an element as part of quick-edit or transfer processing.	A
GENERATE_MODE	F or B	F	Generate the element in Foreground (F) or Batch (B) in Quick-Edit. **Note:** If F, then QE_AUTOGEN cannot be Y.	F
IN_APPROVAL	Y or N	Y	Include packages that are in the approval process.	F
IN_EDIT	Y or N	Y	Include packages that have not yet been cast.	F
IN_EXECUTION	Y or N	Y	Include packages that are executing.	F
INCLUDE_JCL	Y or N	N	Default value for the “INCLUDE JCL” field on the Batch Options panel.	F
INCREMENT_ JOBNAME	Y or N	Y	Indicates whether the Batch Package Facility increments the last character in the jobcard you provide when you submit a package jobstream.	O
INTERCEPT_ISPF_ RETURN	Y or N	N	Instruct CA Endevor SCM whether to intercept ISPF RETURN and JUMP commands.	O

ISPF Dialog Field Name	Valid Values	Current Default	Panel Field Description	Action, Filter, or Other
JCL_PROCEDURE_ NAME	Up to eight char		The name of the CA Endevor SCM procedure that the SUBMIT PACKAGE action will use to create JCL. A sample procedure can be found in iprfx.iqual.CSIQJCL, member name PACKAGE. Use this as a model for your own Submit procedure.	A
LIST_LRECL	0 to 32756	0	Sets the record length default for the LIST DATASET ALLOCATION.	O
MOVE_MODE	F or B	F	Default mode when executing the move action in Quick-Edit.	A
MULTIPLE_ JOBSTREAMS	Y or N	Y	Indicates whether multiple packages are submitted as separate jobs (Y) or as a job steps within a single jobstream (N), when using the Batch Package Facility (ENBP1000).	A
NOSOURCE	Y or N	N	Indicates whether the NOSOURCE option will be used on the GENERATE action.	A
OVERRIDE_ SIGNOUT	Y or N	N	Override the element signout status.	A
PRESERVE_VB	Y or N	Y	Specifies whether saving a variable formatted ISPF edit session preserves trailing spaces in the base element. Effective for CA Endevor Quick-Edit ISPF edit sessions.	O
PROCESSOR_ GROUP_DEFAULT_ CHAR		-	Specify the character to be used to indicate that the processor group symbol is using the default from the PROC statement processor.	O
PROCESSOR_ GROUP_OVERRIDE_ CHAR		O	Specify the character to be used to indicate that the processor symbol has been overridden by the processor group symbol.	O

ISPF Dialog Field Name	Valid Values	Current Default	Panel Field Description	Action, Filter, or Other
PROMOTION	A or P or X	A	Package Foreground Options Menu selection list filter option. If A, lists promotion and non-promotion packages. If P, lists promotion packages only. If X, lists non-promotion packages only.	F
PROMOTION_ PACKAGE	Y or N	N	Defines a package as a promotion package.	A
QE_AUTOGEN	Y or N	N	Quick-Edit ISPF display option. When acting on an element, automatically generate the elements that use this component element. Option for ADD, UPDATE and GENERATE actions only. **Note:** If the value of GENERATE_MODE is F, QE_AUTOGEN cannot be Y.	A
QE_AUTOGEN_SPAN	NONE, ALL, SYS, SUB	NONE	Quick-Edit ISPF display option. To be used with the AUTOGEN option. Automatically generates using elements found in Systems and Subsystems outside the logical map of the AUTOGEN'd component; NONE for no SPAN option, ALL to span all Systems and Subsystems; SYS to span only Systems with the same Subsystem as the component; or SUB to span all Subsystems within the System of the component.	A
QE_BUILD_USING_ MAP	Y or N	Y	Quick-Edit ISPF display option. If set to N, display includes elements in Stage 1 and Stage 2. If set to Y, display includes elements from all stages and environment in map.	
QE_GENERATE_ IN_PLACE	Y or N	N	Quick-Edit ISPF panel action option. If Y, element will be generated in the stage where it is found. If N, element will be copied back into the entry stage.	

ISPF Dialog Field Name	Valid Values	Current Default	Panel Field Description	Action, Filter, or Other
QE_NOSOURCE	Y or N	N	Quick-Edit ISPF display option. Indicates whether the NOSOURCE option will be used on the GENERATE action.	
QE_RETURN_ FIRST_FOUND	Y or N	Y	Quick-Edit ISPF display option. If Y, returns only the first occurrence of the requested elements. If N, returns all occurrences of the requested elements found along the map route.	
REPLACE_MEMBER	Y or N	N	Replace an existing member in a partitioned data set.	A
RETAIN_SIGNOUT	Y or N	N	Retain the signout during move or transfer processing.	A
RETURN_FIRST_ FOUND	Y or N	Y	Include only the first occurrence of the element found in the environment map.	F
SHARABLE_ PACKAGE	Y or N	N	Identify whether a package is sharable.	A
SHOW_TEXT	Y or N	Y	Show text during list action processing.	A
SIGNIN_MODE	F or B	F	Default mode when executing the signin action in Quick-Edit.	A
SIGNOUT_ELEMENT	Y or N	Y	Signout an element during RETRIEVE processing.	A
SIGNOUT_MODE	F or B	F	Default mode when executing the signout action in Quick-Edit.	A
SORT_BY_ DESINATION_ID	1,2 or 3	2	Sort the package shipment list by destination ID.	F
SORT_BY_ PACKAGE_ID	1,2 or 3	3	Sort the package shipment list by package ID.	F
SORT_BY_SHIP_ DATE	1,2 or 3	1	Sort the package shipment list by shipment date.	F
SYNCHRONIZE	Y or N	N	Create a synchronization level as part of move or transfer processing.	A
UPDATE_IF_ PRESENT	Y or N	N	Update an element on an ADD action if the element already exists at the entry stage.	A

ISPF Dialog Field Name	Valid Values	Current Default	Panel Field Description	Action, Filter, or Other
VALIDATE_ACT_ELM_ MASTER	Y or N	Y	Set the default value on the Batch SCL Generation's Validate Element panel. This Validate action option specifies whether to validate that elements were generated correctly.	A
VALIDATE_ACT_SYNC	Y or N	Y	Set the default value on the Batch SCL Generation's Validate Element panel. This Validate action option specifies whether to check for synchronization errors.	A
VALIDATE_ACT_ COMPONENTS	Y or N	Y	Set the default value on the Batch SCL Generation's Validate Element panel. This Validate action option specifies whether to check that all components exist and are valid.	A
VALIDATE_ACT_TERSE	Y or N	N	Set the default value on the Batch SCL Generation's Validate Element panel. This Validate action option specifies whether to limit the amount of message detail that is written to the C1MSGSS1 report file.	A
VALIDATE_ COMPONENTS	Y,N or W	See note	Validate the components of a package as part of cast processing.	A
WITH_HISTORY	Y or N	N	Maintain the element change history.	A
WORKLIST_ PRIMARY	1-999	1	Set the primary defaults for the WORK DATASET ALLOCATION and the LIST DATASET ALLOCATION.	O
WORKLIST_ SECONDARY	1-999	1	Set the secondary defaults for the WORK DATASET ALLOCATION and the LIST DATASET ALLOCATION.	O

Note: For VALIDATE_COMPONENTS the default value depends on the value of the PKGCVAL= parameter in the CA Endevor SCM Defaults Table.

Assemble and Link-Edit the ISPF Configuration Table

Use sample JCL BC1JTABL to assemble and link ENDICNFG to your iprfx.iqual.CSIQAUTU outside of SMP/E or use an SMP/E USERMOD to accomplish this. BC1JTABL is supplied in the installation library iprfx.iqual.CSIQJCL.

Chapter 13: Using the Defaults Table

This section contains the following topics:

The Defaults Table

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

The Defaults Table should reside in an authorized data set.

The C1DEFLTS Macro

There are three occurrences of the C1DEFLTS macro:

- TYPE=MAIN macro defines site-specific information. It must be the first macro specified in the table source. Only one TYPE=MAIN macro can be specified.
- TYPE=ENVRNMNT macro defines environment information. You can have any number of these macros in the table, but you must have one macro for each environment in your configuration. The TYPE=ENVRNMNT macros follow the TYPE=MAIN macro.
- TYPE=END indicates the end of the table definition. It follows the last TYPE=ENVRNMNT macro.

Editing the Defaults Table

When you edit the C1DEFLTS macro, follow the following standard IBM rules:

- Place all macro specifications between columns 2 and 71.
- To continue across lines, place an *X* in position 72 and start the continuation card at position 16 of the next line. Leave at least one blank space between the end of one card (usually a comma) and the continuation character in position 72.
- Include at least one space between the macro name-C1DEFLTS-and the first keyword parameter-TYPE.
- Within each parameter specification, do not include any spaces. If you are using literal operands that include a space, enclose the operand in single quotes.

Important! When editing the macros, pay special attention to the comma and continuation character (X) that follow all but the last parameter within each macro. If either of these is inadvertently deleted, the table assembles with a return code of zero, but problems will result at CA Endevor SCM execution.

Assembling the Defaults Table

After you edit the Defaults Table, you need to assemble and link-edit the table.

You can use an SMP/E USERMOD to assemble and link-edit C1DEFLTS after it has been customized. Alternatively, you can edit the sample JCL BC1JTABL to assemble and link source module C1DEFLTS outside of SMP/E. BC1JTABL is supplied in the installation library *iprfx.iqual.CSIQJCL*. This stores the defaults table in iprfx.iqual.CSIQAUTU as member C1DEFLTS.

The TYPE=MAIN Macro

The TYPE=MAIN Macro defines site-specific information. It must be the first macro specified in the table source. Only one TYPE=MAIN macro can be specified.

The following information shows the TYPE=MAIN macro.

```
C1DEFLTS TYPE=MAIN,                                                    X
               ACCSTBL=,               ACCESS SECURITY TABLE NAME      X
               ACMROOT=,               ACM INDEX ROOT DATA SET NAME    X
               ACMXREF=,               ACM INDEX XREF DATA SET NAME    X
               APRVFLG=N,              APPROVAL PROCESSING (Y/N)       X
               ASCM=N,                 ACM CONTROL OPTION              X
               AUTHTBLS=REQUIRED,      REQUIRED|ALLOW|IGNORE           X
               BATCHID=0,              BATCH UID FROM JOBNAME/USER=    X
               CA7CCINODE=,            CA 7 ADDR SPACE NODE (CAICCI)   X
               CA7JCLID=,              CA 7 JCL DATASET INDEX NUMBER   X
               CA7JCLLIB=,             CA 7 JCL DATASET SYMBOIC INDEX X
               CA7JCLDSN=,             CA7JCLID/CA7JCLLIB DSNAME       X
               CIPODSN=,               CCID VALIDATION DATASET NAME    X
               CNFGTBL=ENDICNFG,       CONFIGURATION TABLE NAME        X
               COMPLISTWD=LIST,        COMPONENT LISTING STRING ID     X
               CUNAME='*** PUT YOUR COMPANY NAME HERE ***', (50 CHAR)  X
               DB2=N,                  DB2 CONTROL OPTION              X
               DESTCFGMBR=,            PARMLIB DEST CONFIG MEMBER      X
               EDITELM=N,              EDIT ELEMENT DIALOG OPTION      X
               ELMCATL='UPRFX.UQUAL.ELMCATL',     E/MVS CATALOG        X
               ELMCATE='UPRFX.UQUAL.ELMCATE',    E/MVS CATALOG EINDEX  X
               ESMTPTBL=ESMTPTBL,      ENDEVOR SMTP TABLE NAME         X
               ESSI=N,                 ESI CONTROL OPTION              X
               EXITTBL=C1UEXITS,       EXIT TABLE NAME                 X
               INFO=N,                 INFO/MAN CONTROL OPTION         X
               JRNLGRP=,               PITR JOURNAL GROUP ID           X
               LIBENV=,                LIBRARIAN (LB), PANVALET (PV)   X
               LIBENVP=N,              LIBRARIAN/PANVALET OPTION       X
               LIBRPGM=,               LIBRARIAN BATCH PROGRAM NAME    X
               LINESPP=60,             LINES PER PAGE                  X
               MACDSN='iprfx.iqual.CSIQOPTN',       E/MVS SOURCE LIBRARY X
               MIXEDFMT=,              ALL|(CCID,COMMENT,DESCRIPTION) X
               MODHLI=,                HLQ FOR TEMPORARY DATA SETS  ) X
               NMAN=N,                 CA Netman INTERFACE OPTION      X
               OPTTBL=ENCOPTBL,        OPTIONS TABLE NAME              X
               PARMLIB=,               ENDEVOR PARMLIB DATA SET NAME   X
               PDM=N,                  PDM CONTROL OPTION              X
               PKGDSN='UPRFX.PACKAGE.MASTER',          PACKAGE DATASET X
               PKGTSO=N,               FOREGROUND PACKAGE EXEC (Y/N)   X
               PKGCSEC=N               PKG CAST SECURITY CHECKING      X
               PKGISEC=N               PKG INSPECT SECURITY CHECKING   X
               PKGCVAL=O,              PKG COMPONENT VALIDATION Y/N/O  X
               PKGSEC=APPROVER,        APPROVER/ESI/MIGRATE            X
               PRBLKSZ=0,              BLKSZE PROC LISTING             X
               PRLNKSZ=(896K,96K),     LINKAGE SIZE FOR PROCS          X
               PRLSTSZ=10,             SIZE ALLOC FOR PROCS LISTINGS   X
               PROC=N,                 PROCESSOR OPTION                X
               RACFUID=,               ALTERNATE ID USERID             X
```

```
RJCLROOT=,              SHIP REMOTE JCL MODEL MBR ROOT X
SITEID=0,               ENDEVOR/MVS SITE IDENTIFIER    X
SMFREC#=0,              SMF RECORD NUMBER              X
SOFETCH=N,              SIGNOUT UPON FETCH       (Y/N) X
SPFEDIT=SPFEDIT,        DEFAULT PDS RESERVE            X
SPAWN=N,                CONCURRENT ACTION PROCESSING   X
SPAWNCNT=,              SPAWN COUNT                    X
SPAWNMAX=,              SPAWN MAXIMUM                  X
SPAWNPROC=,             SPAWN PROCESSOR NAME           X
SYMBOLTBL=,             SITE-DEFINED SYMBOL TBL NAME   X
SYNC@CHK=,              REGRESSION ALERT NOTIFY (Y/N)  X
SYNC@MSV=,              SEVERITY ALERT LVL (I/W/C)     X
SYSIEWL=SYSIEWLP,       DEFAULT PDS/LINK EDIT RESERVE  X
TYPESEQMBR=,            ENDEVOR TYPE SEQ MBR           X
UIDLOC=(1,7),           UID/JOBNAME START/LENGTH POS   X
VIOUNIT=TDISK,          UNIT FOR VIO-ELIGIBLE ALLOC    X
WRKUNIT=TDISK,          UNIT NAME FOR WORK SPACE       X
WORKVOL=                VOL SER NUMBER FOR WRKUNIT
```

The following information defines each TYPE=MAIN parameter.

ACCSTBL

The 1- to 8-character name of the Access Security Table used at your site. You must enter a valid name.

If you are using native security and leave the field blank, no security checking will be done.

If you are using the ESI feature and leave this field blank, the Access Security Table name defaults to **BC1TNEQU**

ACMROOT

The data set name of the VSAM file your site will use to store the name of each CA Endevor SCM Element and all its related components. The recommended name is uprfx.uqual.ACMROOT.

This data set is required if your site is licensed to use the ACM option and have ASCM=Y set in the C1DEFLTS.

Note: For more information, see the *Automated Configuration Option Guide.*

ACMXREF

The data set name of the VSAM file you will use to store the name of each component relationship. The recommended name is uprfx.uqual.ACMXREF.

This data set is required if your site is licensed to use the ACM option and have ASCM=Y set in the C1DEFLTS .

Note: For more information, see the *Automated Configuration Option Guide.*

APRVFLG

Indicates whether approval processing is required. Set this parameter to Y if approval processing is required.

If you set this parameter to N, approval processing will not be active no matter what approver group relationships have been defined.

ASCM

Indicates whether the CA Endevor SCM Automated Configuration Manager (ACM) facility is enabled at your site. If your site is using ACM, specify Y; otherwise specify N.

Note: For more information, see the *Automated Configuration Option Guide.*

AUTHTBLS

Indicates whether CA Endevor SCM's security tables must be loaded from authorized libraries with the values:

REQUIRED

(Default) Authorized libraries are required.

ALLOW

Unauthorized libraries are allowed, but CA Endevor SCM does not check the libraries' authorization.

IGNORE

CA Endevor SCM does not check the libraries' authorization.

BATCHID (Required entry)

Indicates whether the CA Endevor SCM user ID associated with a batch job should be determined by the JOBNAME or the USER parameter specified on the jobcard with the values:

0

(Default) Determined by JOBNAME

1

Determined by the USER parameter

2

Determined by the USER parameter if specified, otherwise by the JOBNAME

Note: For more information, see Selecting a BATCHID Option (see page 228).

CA7CCIDNODE

Specifies the CAICCI node name where the CA 7 Workload Automation address space executes. If no name is specified, local mode is assumed.

CA7JCLID

Provides CA Endevor SCM with the CA 7 Workload Automation parameter information required by CA 7 Workload Automation to schedule JOB execution. Parameter values should be obtained from the CA 7 Workload Automation implementation. Mutually exclusive with CA7JCLLIB.

CA7JCLLIB

Provides CA Endevor SCM with the CA 7 Workload Automation parameter information required by CA 7 Workload Automation to schedule JOB execution. Parameter values should be obtained from the CA 7 Workload Automation implementation. Mutually exclusive with CA7JCLID.

CA7JCLDSN

The data set name associated with CA7JCLID or CA7JCLLIB.

CIPODSN

The data set name of the sequential file your site plans to use to store valid CCID values. The recommended name is uprfx.uqual.VALCCID. This data set is required if you want to use the CCID validation facility.

CNFGTBL

Specifies the name of the Configuration Table. By default, CA Endevor SCM uses the name, ENDICNFG.

COMPLISTWD=LIST

Specifies the string value for identifying listing libraries. You can override this value on the SCL statement PRINT ELEMENT by using the optional clause COMPONENT LIST TEXT STRING. You can override this value on API PRINT ELEMENT requests using the request field ALPRE_RQ_LISTSTR.

CUNAME (Required entry)

The up to 50 character name that describes your site. This name is used in report headings.

DB2

Indicates whether the CA Endevor SCM for DB2 application is enabled at your site. If your site is using CA Endevor SCM for DB2, specify Y, otherwise N.

DESTCFGMBR

Specifies the member name of the destination configuration member in PARMLIB. This member is used by the Post-Ship Script feature to define symbols that are unique by destination for the script files.

EDITELM

Indicates whether the CA Endevor SCM Quick Edit Option is enabled at your site. If your site is using this feature, specify **Y**. Otherwise, enter **N**.

Note: For more information, see the *Quick Edit Option User Guide*.

ELMCATL

The 1-44 character data set name of the Element catalog VSAM file. The Element Catalog supports long Element names and boosts performance by reducing the volume of I/O operations.

ELMCATE

The name of the Element catalog EINDEX VSAM file. If left blank, the name will default to the name specified for ELMCATL with a suffix of .EINDEX.

ESMTPTBL

Specifies the name of the SMTP Table. By default, CA Endevor SCM uses the name, ESMTPTBL.

ESSI

Indicates whether the ESI facility is enabled at your site. If your site is using ESI, specify **Y**. Otherwise, enter **N**.

Important! Do not activate this option until all of the security rules are written and in place. For more information, see the *Security Guide*.

EXITTBL

Specifies the name of the Exit Table. By default, CA Endevor SCM uses the name, C1UEXITS.

INFO

Indicates whether the CA Endevor SCM InfoMan Interface is enabled at your site. If your site is using the interface, specify **Y**. Otherwise, enter **N**. Do *not* activate this option until the InfoMan interface is fully installed.

Note: For more information, see the *Interface for InfoMan Administration Guide*.

JRNLGRP

Specifies a CA L-Serv journal group ID which relates the package data set named in this macro to a specific set of L-Serv journal files. Enables journaling in the package control files and is required for the Point-in-Time Recovery (PITR) feature. The format is: (groupID,nnnn).

groupID

The one- to eight-character Point-In-Time Recovery journal group ID associated with the package journal files.

nnnn

The one- to four-character CA L-Serv journal group subsystem ID.

Note: For more information, see the chapter "Using the Point in Time Recovery Feature" in the *Utilities Guide*.

LIBENV

This parameter serves two purposes depending upon the LIBENVP parameter in C1DEFLTS.

- If LIBENVP=Y, then
  - LIBENV=PV indicates that CA Panvalet is installed;
  - LIBENV=LB indicates that CA Librarian is installed.
- If LIBENVP=N (default), then the include member syntax is used by CONWRITE, by the Expand Include Utility, or by both as follows:
  - LIBENV=PV indicates that checks will be made for CA Panvalet include member syntax (++INCLUDE) or COBOL COPY statements,
  - LIBENV=LB indicates that checks will be made for CA Librarian include member syntax (-inc) or COBOL COPY statements
  - LIBENV=blank (default), *no* syntax checking or expanding will occur.

LIBENVP

Indicates whether your site is using CA Librarian or CA Panvalet with CA Endevor SCM. If your site is using one of these library management applications, specify **Y**. Otherwise, **N**.

LIBRPGM

Applicable only if the LIBENV value is LB. This is the name of the CA Librarian load module for your site.

LINESPP

The number of lines per printed page for output generated by CA Endevor SCM. Default is **60**. Valid entries are 1-99.

MACDSN (Required entry)

The data set name of the library containing the macros. This library was created during the CA Endevor SCM installation procedure. The recommended name is iprfx.iqual.CSIQOPTN. The data set is used by many CA Endevor SCM functions and is required.

MIXEDFMT

Determines whether CA Endevor SCM accepts mixed-case entries in CCID, COMMENT, and DESCRIPTION fields. Values are:

CCID

Accept mixed case in CCID fields.

COMMENT

Accept mixed-case in COMMENT fields.

DESCRIPTION

Accept mixed-case in DESCRIPTION fields.

ALL

Accept mixed-case in all three fields.

NONE

Do not accept mixed-case in any field.

Multiple values can be specified by enclosing the values in () and separating them by a comma. For example, *=(CCID,COMMENT)*.

MODHLI

Indicates the prefix, other than SYSyyddd, that is assigned to a temporary data set, creating a pseudo-temporary data set. The value specified should be a high-level qualifier that all CA Endevor SCM users are authorized to use when allocating, deleting, and opening files for output. If MODHLI is *not specified* in the Defaults Table, the effective name is:

SYS*yyddd*.T*hhmmss*.RA0.*jobname.nnnnnn*

MODHLI has the following uses:

- This parameter applies to those temporary data sets that are allocated with DISP=MOD in any processor step. Regular temporary data sets, which are not DISP=MOD, use the standard z/OS temporary data set name. The modhli prefix appears as the first node of the data set name. The effective name generated is:

  modhli.D*yyddd*.T*hhmmss*.RA0.*jobname*.*nnnnnn*

  modhli

  The value coded on the MODHLI parameter in the C1DEFLTS table.

  D*yyddd*

  Julian date.

  Note: If your site security package requires temporary data sets to follow IBM naming standards, then SYS is used instead of D, so that the node name is: SYS*yyddd*

***T**hhmmss*

Time in hours, minutes, and seconds.

RAO

RA0 is used instead of RA000 to make room for the eight-byte MODHLI and the six-byte temporary data set name.

Note: If your site security package requires temporary data sets to follow IBM naming standards, then the effective name is: SYS*yyddd*.T*hhmmss*.RA000.*ddname*

For more information about the temporary data sets, see the optional feature: DS_INTERNAL_TEMP in the ENCOPTBL table.

jobname

Same value as jobname.

nnnnnn

The temporary data set name &&nnnnnn without the &&.

- This parameter is also used for any client program that accesses the API through Web Services, such as the Eclipse-Based UI, CA CMEW, or a user written client program.
  - If a MODLHI value is coded, your Security administrator must grant all user IDs access to all data sets with the MODHLI high-level qualifier (HLQ). This requirement applies to access under all security products including RACF, CA Top Secret, or CA ACF2.

    The reason for this requirement is as follows. For a client program to access the API, the user ID sent to CAICCI to spawn the pool of STCs and the user IDs that issue requests to Web Services must have read/write access to these data sets. To enable this, the MODHLI parameter causes the data set names to be built with this format:

    modhli.D*yyddd*.T*hhmmss*.STC*nnnnn*.*ddname*

    STC*nnnnn*

    The job ID prefixed by STC.

    ddname

    The unique qualifier for one of the nine API related files (APIMSGS, C1MSGS1, and so on).

  - If a MODLHI value is *not* coded, then security is not affected, because the temporary data sets names are built by the operating system with the standard temporary data set HLQ in the following format: SYSyyddd.Thhmmss.RA000.jobname.nnnnnn

- If your site uses RACF with the RACF PROTECTALL enabled, you must specify a value for MODHLI. When using the Eclipse-Based UI, CA CMEW, or any other client program under RACF with the RACF PROTECTALL option activated, security violations are issued, unless MODHLI is coded. This occurs because the ID used to allocate the temporary files required by Web Services is different than the ID used to open the files. Specifically, the data sets are allocated under the context of the user ID provided on the CA Common Services Common Communications Interface (CAICCI) request to spawn the pool STCs. The open and write requests are issued under the context of the user ID signed on to the client program that is issuing requests through Web Services.

NMAN

Indicates whether the CA Endevor SCM CA Netman Interface is enabled at your site. If your site is using the Interface, specify **Y**. Otherwise, enter **N**.

Note: For more information, see the *CA Netman Interface Administration Guide*.

OPTTBL

Specifies the name of the CA Endevor SCM Options Table. By default, CA Endevor SCM uses the name, ENCOPTBL.

PARMLIB

Indicates the Parmlib data set name, which is required to implement the Global Type Sequencing option, because it contains the Type Sequence member.

PDM

Indicates whether the CA Endevor SCM Parallel Development (PDM) facility is enabled at your site. If your site is using PDM, specify **Y**. Otherwise, enter **N**.

Note: For more information, see the *PDM User Guide*.

PKGDSN

(Required entry if you are using packages) The data set name of the VSAM file your site will use to store package information. This library was created during the installation procedure. The recommended name is uprfx.uqual.PACKAGE.

The data set is required if your site plans to use any of the package processing features.

Note: For more information, see the *Packages Guide*.

Be sure you use the same data set name for both definition of and allocation of the package data set.

PKGTSO

Indicates whether foreground package processing is allowed. Set this parameter to **Y** if foreground processing is allowed at your site. Otherwise, set the parameter to **N**.

When a processor runs as part of a package, the PKGTSO flag overrides the foreground execution flag in the processor group. This means that if PKGTSO=**Y**, all processors in that package will execute, regardless if the value in the foreground execution field for the processor group.

PKGCSEC

Indicates whether actions should incur a security check at package cast time. Set this parameter to **Y** to have each CA Endevor SCM action checked during the package cast operation.

If you set this parameter to **N**, no action security check is performed during the package cast operation.

PKGISEC

Indicates whether actions should incur a security check at package inspect time. PKGISEC does not have to be defined. If omitted, CA Endevor SCM uses the PKGCSEC value to determine whether security checks are done during the inspect action. Set this parameter to **Y** to have each CA Endevor SCM action checked during the package inspect operation.

If you set this parameter to **N**, no action security check is performed during the package inspect operation.

PKGCVAL

Indicates whether component validation is required when packages are cast. Values are:

Y

Component validation is required, no matter who is working with the package.

O

Component validation is optional. Whether validation takes place is determined by the person working with the package.

PKGSEC

Indicates the type of security which controls the package options.

APPROVER

This value indicates that the site would like to restrict package actions to package approvers.

ESI

Indicates that the site would like to control package options through an external security package such as CA ACF/2 for z/SO, CA Top Secret, and IBM RACF via the ESI interface.

MIGRATE

Indicates that the site is in transition between Approver security and ESI security. Both will be checked.

Note: The approver security rules take precedence over ESI security rules. If the user is granted access to the package by the approver rules, ESI will not be invoked. ESI will be invoked only when the user does not belong to any approver groups associated with the package (If there are no approver groups associated with the package (this is true for ALL packages before they are CAST), no access restrictions apply.)

PRBLKSZ=0

Indicates the blocksize to be used for temporary files when compiling processors. If you are on a version of z/OS that does not support BLKSIZE=0, you must specify the value for this parameter that is a multiple of 121.

PRLNKSZ=(256K,64K)

Indicates the size parameter used by the linkage editor. When adding very large processors with many symbolics, this parameter may need to be increased. For example, use (384K,64K).

PRLSTSZ=10

The value of PRLSTSZ is used for both the primary and secondary space allocation parameters (in cylinders) for processor listings. The value of PRLSTSZ may need to be increased for very large processors.

PROC

Indicates whether processors are enabled at your site. If your site is using processors, specify **Y**. Otherwise, enter **N**.

Note: For more information, see the *Extended Processors Guide*.

RACFUID

The alternate user ID for data set authorization checking.

RJCLROOT

Controls the choice of remote JCL generation. There are three choices:

RJCLROOT

Not specified. Causes the job stream to be programmatically generated without the use of model members.

RJCLROOT=FCPY

Specifies the job stream is generated through a set of model members beginning with the character string #RJFCPY, in order to build CA-PDSMAN FASTCOPY JCL for the remote copy and deletes.

RJCLROOT=ICPY

Specifies the job stream is generated through a set of model members beginning with the character string #RJICPY. These members rebuild IEBCOPY and IDCAMS JCL statements.

SITEID (Required entry)

The 1-character name that identifies your site. It is used internally to differentiate between sites. Default is **0** (zero).

SITEID is an integral part of the CA Endevor SCM footprint. Any changes to this parameter for existing CA Endevor SCM installations will result in a footprint-compromised error for each Element within that installation's environments.

SMFREC#

The SMF record number assigned to SMF records written by CA Endevor SCM at this site. Set this value to *0* (zero) until you implement the SMF option.

If you want to write out SMF activity records or SMF security records or both, you must enter a value for this parameter. The parameters controlling which records are actually written are SMFACT and SMFSEC in the TYPE=ENVRNMT macro.

SOFETCH

Indicates whether the Element that is fetched should be signed out to you, if not already signed out to someone else **Y**, or not signed out **N**.

This value will come into play with Add (Fetch), Generate (Copyback), Move (Fetch), Transfer (Fetch), Search and Replace (Fetch) and Quick-Edit.

SPFEDIT

The queue name used when CA Endevor SCM issues an enqueue on a sequential file or partitioned data set (not RECFM=U). This may be a source library, object library, or other user library. Default is **SPFEDIT**.

The resource name for the enqueue name is the data set name.

SPAWN

Determines if the Concurrent Action Processing Feature is enabled. Allowable values are Y or N.

N (default) – Indicates that concurrent action processing is disabled. Actions will be processed one at a time regardless of any other C1DEFLTS parameters or the presence of the EN$CAP or EN$CAPnn JCL DD cards on the job request JCL.

Y – Indicates that Concurrent Action Processing is enabled. However, if SPAWNCNT=0 then concurrent action processing will not take place unless an EN$CAP*nn* DD card is present in the JCL and *nn* (number of servers to spawn) is greater than 2.

SPAWNCNT

The number of action request regions allowed for a job. Allowable values are 0 and 2-99. The value must be less than or equal to the SPAWNMAX value.

0 –Concurrent Action Processing is disabled. Actions are single threaded, regardless of the setting of SPAWN.

2-nx –Indicates the maximum number of action request regions allowed per task. A value greater than zero is not allowed if SPAWN=N. This is the number of processors that are created if the EN$CAP DD card is present in the JCL.

Note: If SPAWN=Y is specified, then the user can override a SPAWNCNT of 0 by using a EN$CAPnn DD card in the job request JCL. If the nn value in the EN$CAPnn JCL DD statement is 00, then Concurrent Action Processing will be disabled for the job.

SPAWNMAX

The maximum number of action request regions allowed for a job. Allowable values are 2-99. Note that zero is valid if the SPAWN indicator is set to a value of N.

SPAWNPROC

The one to eight-character name of the Concurrent Action Processing server. This name should be a CA CCI SERVICE definition service_name and a started task procedure name. A name is only required if the SPAWN indicator is set to a value of Y.

SYMBOLTBL

Indicates the name of the symbolics table in use at your site if you are using site-defined symbolics.

Note: For more information, see Site-Defined Symbolics (see page 153).

SYNC@CHK

Specifies whether source synchronization is active at a site. Valid values are Y or N or blank. The default is: SYNC@CHK=,

A blank or a N specification deactivates Source synchronization.

SYNC@MSV

When source is found out-of-sync with an instance of itself at a higher location (which is marked as a sync location) in the software lifecycle, log messages are written to the action log. This parameter determines the severity of these messages. An *I* specification will produce Informational out-of-sync log messages, a *W* produces Warning log messages and a *C* produces Caution log messages. The default is: SYNC@MSV=,

A blank specification will default to *I*.

SYSIEWL

The queue name used when CA Endevor SCM issues an enqueue on a partitioned data set defined with RECFM=U (for example, a load library). Default is **SYSIEWLP**.

The resource name for the enqueue is the data set name.

TYPESEQMBR

Indicates the name of the Type Sequence member, which enables Global Type Sequencing. The Type Sequence member defines the type sequence order in which Element actions are processed at your site, regardless of the action's inventory location. If you specify this parameter, you must also specify the Parmlib data set parameter. An empty Type Sequence member will force type sequencing by type name order.

UIDLOC

The character positions of the user ID that should be compared for ownership (that is, for Element signout and signout override check) in batch. You specify a starting position and number of characters.

For example, if you wanted to take the first three characters of the user ID, then you would specify UIDLOC=(1,3). Default is the first seven characters: UIDLOC=(1,7).

VIOUNIT (Required entry)

The unit name for temporary disk data sets that are stored on a virtual I/O unit.

WRKUNIT (Required entry)

The unit name for temporary disk data sets that are *not* stored on a virtual I/O unit.

WORKVOL

The volume serial number of the disk used to store temporary data sets.

The TYPE=ENVRNMNT Macro

The TYPE=ENVRNMNT macro defines environment information. You can have any number of these macros in the table, but you must have one macro for each environment in your configuration. The TYPE=ENVRNMNT macros follow the TYPE=MAIN macro.

The following is the TYPE=ENVRNMNT macro.

```
C1DEFLTS TYPE=ENVRNMNT,                                          X
      ENDBAVL=N,               ENDEVOR DB OPTION AVAILABLE     X
      ENDBACT=N,               ENDEVOR DB ACTIVE FLAG          X
      ENTRYSTG#=1,             ENTRY STAGE IS STAGE 1          X
      ENVNAME='MVSTEST',       ENVIRONMENT NAME (8 CHAR)       X
      ENVTITL='SAMPLE TEST ENVIRONMENT', TITLE (40 CHAR)       X
      JRNLGRP=,                PITR JOURNAL GROUP ID           X
      NEXTENV=(MVSPROD,P),     NEXT ENV/STG ID IN MAP (8,1)    X
      RSCETBL=,                RESOURCE SECURITY TABLE NAME    X
      SMFACT=N,                SMF ACTIVITY OPTION (Y/N)       X
      SMFSEC=N,                SMF SECURITY OPTION (Y/N)       X
      STG1ID='U',                                              X
      STG1NME='UT',                                            X
      STG1PSAS=N,              STAGE 1 PROM STOP AT STG (Y/N)  X
      STG1TTL='UNIT TEST',                                     X
      STG1VSM='SPRFX.SQUALUT.MCF',                             X
      STG2ID='Q',                                              X
      STG2NME='QA',                                            X
      STG2PSAS=N,              STAGE 2 PROM STOP AT STG (Y/N)  X
      STG2TTL='QUALITY ASSURANCE',                             X
      STG2VSM='SPRFX.SQUALQA.MCF',                             X
      SYNC@BLOC=N,             REGRESSION ALERT START LOC      X
USERTBL=                 USER SECURITY TABLE NAME
C1DEFLTS TYPE=ENVRNMNT,                                          X
      ENDBAVL=N,               ENDEVOR DB OPTION AVAILABLE     X
      ENDBACT=N,               ENDEVOR DB ACTIVE FLAG          X
      ENTRYSTG#=1,             ENTRY STAGE IS STAGE 1          X
      ENVNAME='MVSPROD',       ENVIRONMENT NAME (8 CHAR)       X
      ENVTITL='SAMPLE PRODUCTION ENVIRONMENT', TITLE (40 CHAR)X
      JRNLGRP=,                PITR JOURNAL GROUP ID           X
      NEXTENV=,                NEXT ENV/STG ID IN MAP (8,1)    X
      RSCETBL=,                RESOURCE SECURITY TABLE NAME    X
      SMFACT=N,                SMF ACTIVITY OPTION (Y/N)       X
      SMFSEC=N,                SMF SECURITY OPTION (Y/N)       X
      STG1ID='I',              STAGE 1 IDENTIFIER (1 CHAR)     X
      STG1NME='INT',           STAGE 1 NAME (8 CHAR)           X
      STG1PSAS=N,              STAGE 1 PROM STOP AT STG (Y/N)  X
      STG1TTL='INTEGRATION STAGE',  STAGE 1 TITLE (20 CHAR)    X
      STG1VSM='SPRFX.SQUALINT.MCF',       STAGE 1 MCF (VSAM)   X
```

```
      STG2ID='P',              STAGE 2 IDENTIFIER (1 CHAR)    X
      STG2NME='PRD',           STAGE 2 NAME (8 CHAR)          X
      STG2PSAS=N,              STAGE 2 PROM STOP AT STG (Y/N) X
      STG2TTL='PRODUCTION STAGE',  STAGE 2 TITLE (20 CHAR)    X
      STG2VSM='SPRFX.SQUALPRD.MCF',       STAGE 2 MCF (VSAM)  X
      SYNC@BLOC=N,             REGRESSION ALERT START LOC     X
USERTBL=                 USER SECURITY TABLE NAME
```

To specify more than one environment, copy this macro as many times as necessary.

Note: If you are creating an environment map, you need to specify the next environment in each definition. Make sure that you code them in the proper order; starting with the first environment and continuing to the last environment (in other words, start with the top and work down).

The following shows an example for three environments:

```
     C1DEFLTS TYPE=ENVRNMNT,                  X
              .
              .
              .
        USERTBL=
     C1DEFLTS TYPE=ENVRNMNT,                  X
              .
              .
              .
        USERTBL=
     C1DEFLTS TYPE=ENVRNMNT,                  X
              .
              .
              .
        USERTBL=
```

The following information defines each TYPE=ENVRNMNT macro.

ENDBACT ENDBAVL

Parameters related to the CA Endevor SCM to CA Endevor/DB for IDMS Bridge.

ENTRYSTG#

Indicates the entry stage, either 1 or 2, for the environment. If this parameter is not defined, the entry stage defaults to 1.

ENVNAME (Required entry)

The up to eight character name of the CA Endevor SCM environment to which this macro applies. This value must be unique within this Defaults Table (that is, to this site). The name can include any of (but only) the following characters: A-Z, 0-9, @, #, $.

ENVNAME is an integral part of the CA Endevor SCM footprint. Any change to this parameter for existing z/OS environments will result in a footprint-compromised error for each Element within that environment.

ENVTITL (Required entry)

The up to 40 character descriptive title for the environment.

JRNLGRP

An L-Serv journal group ID which, when entered, relates the Master Control Files named in this macro to a specific set of L-Serv journal files. Use of this parameter enables journaling in the Master Control File base and delta libraries (both VSAM and non-VSAM).

The format is: (gggg,nnnn)

gggg

The journal group ID associated with the MCF and delta journal files.

nnnn

The journal group subsystem ID.

Note: For more information, see the *Utilities Guide.*

NEXTENV

The next environment/stage location on the environment map. The format is: (*environment name, stage id*)

environment name

The 1- to 8-character name of the next environment.

stage id

The identifier (1 or 2) of the stage in that environment. If you do not provide a stage ID, CA Endevor SCM defaults to STAGE id=1.

RSCETBL

The 1- to 8-character name of the CA Endevor SCM Resource Security Table for this environment. The Resource Security Table allows you to restrict one or more Elements to a specific system(s) and subsystem(s) within an environment. Define one Resource Security Table for each environment (as necessary).

Leave this field blank if you do not want to restrict Elements to a specific system(s) and subsystem(s). If you do want to use restriction, fill in this field and prepare the Resource Security Table as described in the *Security Guide.* (only applies to native security).

SMFACT

Indicates whether CA Endevor SCM should write out SMF activity records. Set this parameter to **Y** if you want activity records written out. Otherwise, set the parameter to **N**. Default is **N**.

If you set this parameter to **Y**, you must specify a value for the SMFREC# in the TYPE=MAIN macro. If you do not specify SMFREC#, CA Endevor SCM ignores the SMFACT parameter.

SMFSEC

Indicates whether CA Endevor SCM should write out SMF security records. Set this parameter to **Y** if you want security records written out. Otherwise, set the parameter to **N**. Default is **N**.

If you set this parameter to **Y**, you must specify a value for the SMFREC# in the TYPE=MAIN macro. If you do not specify SMFREC#, CA Endevor SCM ignores the SMFSEC parameter.

STG1ID (Required entry)

The 1-character identifier for the first stage, used during processing to select (or identify) the stage you want. Commonly used identifiers are 1 or A. Default is **1**.

STG1NME (Required entry)

The up to eight character name assigned to Stage 1 for this environment. The stage name must be unique across all environments within a site (that is, within the Defaults Table). The stage name can include any of (but only) the following characters: A-Z, 0-9, @, #, $.

Some commonly used Stage 1 names are: PREPROD, QA, STAGE1ID, and TEST.

STG1PSAS

This is a stage level parameter that, if set to Y, activates the stop-at-stage feature for promotion packages. You can activate this feature for one or both stages within the same environment. To activate this feature for a second stage, use STG2PSAS. You can also activate this feature in more than one environment (for example, in both QA/1 and PRD/1).

If you do not wish to activate this feature you can code “STG1PSAS=N”, or “STG1PSAS=”, or not code the parameter at all. The default is N.

STG1TTL (Required entry)

The 1- to 20-character descriptive title for the first stage. This name is used on various panels and reports to describe this stage.

STG1VSM (Required entry)

The Master Control File data set name for Stage 1, as defined during installation. The data set name defaults to: uprfx.uqual.STAGE1.

STG2ID (Required entry)

The 1-character identifier for the second stage, used during processing to select (or identify) the stage you want. Commonly used identifiers are 2 or B. Default is **2**.

STG2NME (Required entry)

The up to eight character name assigned to Stage 2 for this environment. The stage name must be unique across all environments within a site (that is, within the Defaults Table). The stage name can include any of (but only) the following characters: A-Z, 0-9, @, #, $

STG2PSAS

This is a stage level parameter that, if set to Y, activates the stop-at-stage feature for promotion packages at a second stage. You can activate this feature for one or both stages within the same environment. To activate this feature for the first stage, use STG1PSAS. You can also activate this feature in more than one environment (for example, in both QA/1 and PRD/1).

If you do not wish to activate this feature you can code “STG2PSAS=N”, or “STG2PSAS=”, or not code the parameter at all. The default is N.

STG2TTL (Required entry)

The 1- to 20-character descriptive title for the second stage. This name is used on various panels and reports to describe this stage.

STG2VSM (Required entry)

The Master Control File data set name for Stage 2, as defined during installation. The data set name defaults to: uprfx.uqual.STAGE2.

SYNC@BLOC

Specifies whether the Environment, Stage location is a source synchronization location. If SYNC@BLOC=Y, when an action adds or moves an Element to this location, or to any location higher up the map, a sync check is performed. If an Element located lower in the map is not in sync with the new Element, a synchronization notification email is sent to the owner of the out-of-sync Element at the lower location. If more than one SYNC@BLOC is specified in C1DEFLTS, regardless of its specification (SYNC@BLOC=, SYNC@BLOC=N, or SYNC@BLOC=Y,) all SYNC@BLOC parameters after the first one in the map are ignored.

USERTBL

The up to eight character name of the CA Endevor SCM User Security Table for this environment. The User Security Table allows you to control the systems and subsystems available to all users within an environment. Define one User Security Table for each environment (as necessary).

Leave this field blank if you do not want to control systems and subsystems within an environment. If you do want to control the availability of systems and subsystems, fill in this field and prepare the User Security Table as described in the *Security Guide.* (only applies to native security).

Selecting a BATCHID Option

When executing CA Endevor SCM element actions in batch (C1BM3000), CA Endevor SCM uses the values specified for BATCHID and UIDLOC in the C1DEFLTS table to derive the value to be used as the CA Endevor SCM userid.

The BATCHID parameter is specified in the TYPE=MAIN section of your C1DEFLTS table. It determines whether the CA Endevor SCM userid associated with a batch job will be determined by the JOBNAME, the USER= parameter specified on the jobcard the userid of the submitter. One of the following values must be specified:

- 0-determined by JOBNAME (this is the current default value, future releases will default to 2).
- 1-determined by the USER parameter, or if not present, the userid of the submitter.
- 2-determined by the USER parameter (or submitter as described previously) if specified, otherwise by the JOBNAME.

The UIDLOC parameter of the C1DEFLTS table (also TYPE=MAIN), tells us the character positions of the CA Endevor SCM userid that should be compared for ownership, (that is, for element signout/override signout check) in batch. UIDLOC specifies the starting position and the number of characters. The default is the first seven characters UIDLOC=(1,7).

In the following example, user PAND and user NDVR attempt to RETRIEVE an element that is currently "owned" by user NDVR. UIDLOC=(1,4). Neither user has the authority to use the "OVERRIDE SIGNOUT" option. The following information shows the results of their attempts:

		Userid Assigned by Endevor/Action Allowed (Y/N)			
JOBNAME	SUBMITTER	USER= (1)	BATCHID=0 (2)	BATCHID=1	BATCHID=2
ABCDJ	PAND	PAND	ABCD/N	PAND/N	PAND/N
ABCDJ	PAND	n/a	ABCD/N	PAND/N	PAND/N
ABCDJ	NDVR	NDVR	ABCD/N	NDVR/Y	NDVR/Y
ABCDJ	NDVR	n/a	ABCD/N	NDVR/Y	NDVR/Y
PANDJ	PAND	PAND	PAND/N	PAND/N	PAND/N
PANDJ	PAND	n/a	PAND/N	PAND/N	PAND/N
PANDJ	NDVR	NDVR	PAND/N	NDVR/Y	NDVR/Y
PANDJ	NDVR	n/a	PAND/N	NDVR/Y	NDVR/Y
NDVRJ	PAND	PAND	NDVR/Y(2)	PAND/N	PAND/N
NDVRJ	PAND	n/a	NDVR/Y(2)	PAND/N	PAND/N
NDVRJ	NDVR	NDVR	NDVR/Y	NDVR/Y	NDVR/Y
NDVRJ	NDVR	n/a	NDVR/Y	NDVR/Y	NDVR/Y

(1) Several security packages have an optional feature to automatically append the user=/password= parameters on the job card at job submission. If this is the case at your shop, then BATCHID=1 is recommended.

(2) Many sites have job submission exits that ensure the jobname matches the submitter's userid. If your site does not utilize this feature, it is strongly recommended you DO NOT USE BATCHID=0.

The TYPE=END Macro

The TYPE=END macro indicates the end of the table definition. It follows the last TYPE=ENVRNMNT macro.

The TYPE=END macro is shown next:

```
C1DEFLTS TYPE=END
```

Do not change this macro.

Chapter 14: Using Alternate Customization Tables

This section contains the following topics:

Alternate Customization Tables

You can use the ENUXSITE exit to select an alternate Defaults Table (C1DEFLTS) based on a user ID or other criteria.

If your site has many different groups that use the same libraries, but each group uses a different C1DEFLTS table, using alternate customization table names can make it easier to manage these groups.

You can identify the following tables within the C1DEFLTS Table:

- Optional Features Table (ENCOPTBL)
- Configuration Table (ENDICNFG)
- SMTP EMAIL Table (ESMTPTBL)
- User Exits Table (C1UEXITS)

Through the use of the ENUXSITE exit, unique C1DEFLTS tables can identify the table names that are to be loaded enabling flexibility for administrators.

The ENUXSITE Program

ENUXSITE is an exit, called at CA Endevor SCM initialization that allows your site to identify the name of an alternate Defaults Table (C1DEFLTS) based on a user ID or other criteria. An alternate Defaults Table might be used as a mechanism to test a new configuration at your site. Or, you may want to provide a specific Defaults Table for a group of users based on particular installation criteria.

Note: The ENUXSITE program must be linked as an authorized program and must reside in an authorized library. In addition, the alternate Defaults Table must exist before you implement the exit.

ENUXSITE Program Parameters

When CA Endevor SCM initializes a session, it searches the authorized library for a user-written program named ENUXSITE. If the program is found, CA Endevor SCM passes to the exit, using standard linkage conventions, a single, 16-character data field. The first eight characters of the field contain the name C1DEFLTS. The remaining eight characters contain the user ID associated with the current CA Endevor SCM session.

When running in batch, the value specified as the user ID will be either one of the following:

- The job name specified on the jobcard.
- The value specified on the USER= parameter of the jobcard. This value, if provided, takes precedence over use of the job name.

Using the ENUXSITE Program

There are two ways to select an alternate Defaults Table:

- Modify the parameter by changing the first eight characters to the name of the Defaults Table you want to use. ENUXSITE returns control to CA Endevor SCM to load the named table and use it as the CA Endevor SCM Defaults Table.
- Use the z/OS LOAD service to load an alternate table. Use the IDENTIFY service to identify the table as the “C1DEFLTS.” The exit returns control to CA Endevor SCM, leaving the 16-character parameter unmodified.

Creating an Alternate Defaults Table

To create an alternate Defaults Table, follow the instructions for creating the Defaults Table as provided in the implementation instructions. Change the name C1DEFLTS to the name you want to assign the alternate Defaults Table. Verify that the link-edit output member name--DD SYSLMOD--specifies the correct name.

Example: Use a Different Version of the Defaults Table

The following example shows a COBOL program written to allow USER1 to have a different version of the Defaults Table than other users. The name of USER1's Defaults Table is TABLE1.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. ENUXSITE.
ENVIRONMENT DIVISION.
DATA DIVISION.
LINKAGE SECTION.
01 LS-PARM-FROM-CA ENDEVOR.
  05 LS-TABLE-NAME     PIC X(8).
  05 LS-USER-ID      PIC X(8).
PROCEDURE DIVISION USING LS-PARM-FROM-CA ENDEVOR.
  IF  LS-USER-ID = 'USER1  '
  THEN MOVE 'TABLE1 ' TO LS-TABLE-NAME.
  GOBACK.
```

Chapter 15: Performance and Tuning

This section contains the following topics:

Forward and Reverse Delta Formats

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

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

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

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

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

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

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

The following table summarizes the forward and reverse delta formats.

Delta Type	I/O operations needed-read	I/O operations needed-write	Use for elements that:
Forward	2	1	■ Are updated more than read ■ Are in production.
Reverse	1	2	Normally needed a source output library but do not need to be backed out. These files need to be kept in a standard format (such as a PDS) for utilities such as Advantage CA File Master.

Note: For more information about the conversion process, see the appendix "Converting Delta Formats."

Delta Level Management

CA Endevor SCM provides three methods to consolidate delta levels: Automated Element Level Versioning; Auto Consolidation; and Aged Delta Retention. Automated Element Level Versioning is the default, but it is also used in conjunction with the other two methods, both of which are optional. Aged Delta Retention is managed at the system level; the others are managed at the type level. These methods are summarized in the following table:

Delta Level Management Method	Level Where Implemented	Implementation Effort	Effect
Automated Element Level Versioning	Type	Default	Up to 9996 delta levels are retained.
Auto Consolidation	Type	Optional	■ You set the number of levels to be retained up to a maximum 96 levels. ■ You set the number of levels that are consolidated when the number of levels to be retained is reached.
Aged Delta Retention	System	Optional	You set the number of months that delta levels are retained up to a maximum 999 months.

Automated Element Level Versioning

Automated Element Level Versioning increases the limit on delta levels to 9996 and is activated at the Type level.

For each Type to be activated, navigate to the Type Definition panel in foreground and set AUTO CONSOL to N and LVLS TO CONSOL to 0. If you were not previously using Auto Consolidation, and had not changed these fields, then these values will already be set.

Alternatively, you can run the Batch Admin DEFINE TYPE action to set these field values for each Type to be activated. With Auto Consolidation turned off, you may eventually run out of levels or accumulate more levels than you really need.

Element Delta Consolidation Method

Auto consolidation is implemented by type and lets you specify the physical level threshold at which the consolidation process is initiated and the number of physical levels to consolidate when the threshold is met. The maximum number of levels you can keep is 96.

Element data consolidation consists of the following two functions:

- **Consolidation level (CONSOL AT LVL)-**specifies when the consolidation process is initiated. This is the maximum number of physical levels you want stored on the system. For example, the CONSOL AT LVL parameter is set to 96. When the physical number of levels reaches 97, CA Endevor SCM begins the consolidation process.

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

- **Number of levels to consolidate (LVLS TO CONSOL)-**specifies the number of physical levels to consolidate when the threshold is met. For example, suppose the LVLS TO CONSOL parameter is set to 25, and the CONSOL AT LVL (consolidation trigger) is set to 96. An element in Production is at the maximum level 0196, and vvll 0197 is about to be moved in. Prior to saving the 97th level, CA Endevor SCM does the following:

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

    CONSOL AT LVL + 1 97

    —LVLS TO CONSOL: — 25

    Minimum level 72

  When working in the lower stages where the level number is not equal to the physical number of levels, Auto Level Versioning is used to increment or “roll over” the vvll value until the number of physical levels specified on CONSOL AT LVL is reached.

Using the same consolidation values as in the prior example, suppose an element at vvll 0194 is retrieved from the production stage for modification. When it is added back into CA Endevor SCM at a lower stage, level 0194 is fetched back for compare and level 0195 is created. The level number is 95, but the number of physical levels is two-- 0194 and 0195. For auto consolidation to be triggered in this lower stage, 94 more changes would have to be made to get up to 96 physical levels. Auto versioning comes into play when vvll 0199 is reached, the next change would roll over to 0200, then 0201 and so on until 0290 is reached (0194+96=0290), should one more change be attempted auto consolidation would be triggered causing the lower 25 levels, 0195 through 0220, to become a new 0195, and levels 0221 through 0290 to be renumbered accordingly.

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

Maximum level (CONSOL AT LVL)

—Minimum level

Levels to consolidate (LVLS TO CONSOL)

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

Note: Verify the parameter values match for the same element type across different CA Endevor SCM locations.

For more information about the type definition fields LVLS TO CONSOL and CONSOL AT LEVEL, see Type Definition Fields in the chapter Defining Inventory Structures.

Aged Delta Retention Method

The Aged Delta Retention method lets you set the number of months that delta levels are retained. The maximum retention limit is 999 months. This option is inactive by default and is set on each system definition in foreground or batch. It can be set for elements, components, or both, and overrides any consolidation levels set on all type definitions in that system. This method is invoked only at the end of the mapped system location route.

When auto age level retention is set, change levels for all types in the system expire after the number of months specified on the system definition. The base element or component is rebuilt to reflect the changes from the deleted levels, with the base delta version and level numbers set to the last discarded version and level. The delta format you are using, reverse, forward, or full image, determines how the consolidated level is built. For reverse and full image deltas, change levels that expire are dropped. For forward deltas, change levels that expire are incorporated into the base.

Example: Aged Delta Retention

In this example, the delta levels are set to expire after five months, and the element is changed and moved through the lifecycle as follows:

- The element named ELEMENT1 is added to stage 1 of the Test environment from stage 2 of the Production environment. The highest level of the element in stage 2 of the Production environment, which is 0103, is fetched to stage 1 of the Test environment.
- The element is updated in stage 1, Test, four times over a ten month period. Each time the element is saved, a new level is created with the level number incremented by 01. So that after the element is saved four times, the new levels are 0104, 0105, 0106, and 0107.
- The element is moved with history to stage 2, Test. All the levels in stage 1, Test (0103, 0104, 0105, 0106, and 0107) are included in the move.
- The element is moved with history to stage 2 of the Production environment. Auto age level retention is triggered because the element has reached the end of the map and some of the element levels are older than the five month retention period. The result of the move with history to stage 2, Production, depends on the delta format as described next:

  If reverse delta (the base is a full copy of the current code):

  - Levels 1 thru 4 are removed, because they are older than 5 months.
  - Level 4 becomes the new base level 00.
  - A new base delta record is created reflecting level 4 with the following modifications:
    - All level 4 change statements are removed.
    - Insert and delete record counts are set to zero.

  If forward delta (the base is a full copy of the original code):

  - Base is re-built to contain change levels 1 thru 4, because they are older than 5 months.
  - Level 4 becomes the new base level 00.
  - A new base delta record is created. The base level 0 record reflects level 4 with the following modifications:
    - All level 4 change statements are removed.
    - Insert and delete record counts are set to zero.

If full image delta

- Levels 1 thru 4 are removed, because they are older than 5 months. The full image deltas for level 0 through level 3 are deleted.
- A new base level record is created. The base level 0 record reflects level 4.

Note: Regardless of the delta level format, the delta's main control record (the first record in the file) is updated to reflect the new base level. In this example, level 4 is made into the new base level 0. The base record count, date and time fields reflect level 4.The last level value and record count reflect level 4. The last level date and time fields are set to blanks.

The following chart illustrates this example and shows how delta levels are incremented and consolidated using the auto age delta retention feature.

Env: Test Stg: 1	**Env: Test Stg: 2**	**Env: Prod Stg: 2**
		ElEMENT1 exists with 3 levels: ELEMENT1 (0100) (0101) (0102) (0103)
First add of ELEMENT1 causes a fetch of (0103) with a new delta level to record changes. ELEMENT1 (0103) (0104)		
Delta Age Retention is active and set to 5 months. Changes are made over a 10 month period. ELEMENT1 (0103) (0104) (0105) (0106) (0107) Delta management will not be done since the element hasn't reached the end of its route.		
	Results of Move with history: ELEMENT1 (0103) (0104) (0105) (0106) (0107)	
		Results of Move with history: Delta management is triggered, because levels 1 thru 4 are older than the 5 month retention limit. ELEMENT1 (0100) (0101) (0102) (0103) **See note.**

Note: In this example, at the end of the map, if the level format is:

- *reverse delta*, then levels 1 thru 4 are removed. Level 4 becomes the new base level 00.
- *forward delta*, then the base is re-built to contain change levels 1 thru 4. Level 4 becomes the new base level 00.
- *full image delta*, then levels 1 thru 4 are removed.

Implement Age Level Retention

To retain delta levels for a specific number of months, you can activate the Age Level Retention feature for each system in your environment. This may help you to meet record keeping requirements for your organization or regulatory requirements. To activate this delta level management method in foreground, you can update the system definition through the online environment update facility. You can activate Age Level Retention for elements, components, or both.

Note: By default, the age delta level retention options on all the system definitions are turned off for both elements and components. The default settings are ELEMENT=N and COMPONENT=N and the only allowable value for their corresponding RETAIN LVLS FOR fields is 0.

To implement Age Level Retention

1. On the System Request panel, select option U.

   The Update System Definition panel opens. The following options appear in on this panel in the AUTO AGE LEVEL RETENTION OPTIONS fields.

2. (Optional) Set the ELEMENT option =Y.

   This activates Age Level Retention for elements of all types in the system.

3. To specify how many months the element delta levels will be retained, type the number of months in the RETAIN LVLS FOR field that appears next to the ELEMENT option. Allowable values are 1 to 999 and the default is 999.

   Element delta levels will be retained for the number of months specified, provided the ELEMENT option is set to Y.

4. (Optional) Select Y for the COMPONENT option.

   This activates Age Level Retention for components of all types in the system.

5. To specify how many months the element delta levels will be retained, type the number of months in the RETAIN LVLS FOR field that appears next to the COMPONENT option. Allowable values are 1 to 999 and the default is 999.

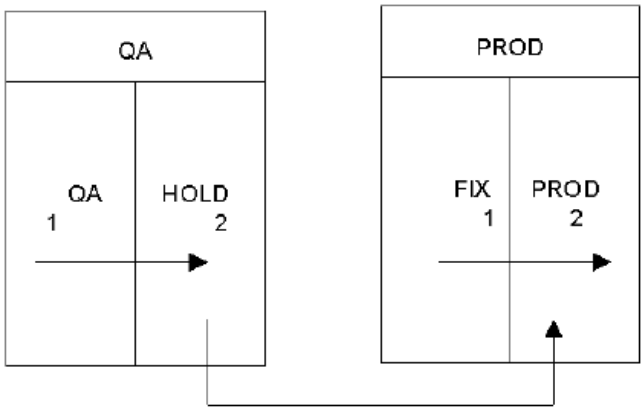
   Component delta levels will be retained for the number of months specified, provided the COMPONENT option is set to Y.

Note: After you set retention limits in the system definitions, you can examine what effect this will have on elements and components by using the Age-Managed Delta utility, in examine mode, to see which delta levels have expired. For more information, see the chapter "Using the Age-Managed Delta Utility" in the *Utilities Guide*.

Mapping Multiple Environments

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

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



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

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

Selecting a Library Type for Base and Delta Members

You can store your CA Endevor SCM data sets using PDS, PDS/E, ELIB (VSAM/BDAM), or CA Panvalet and CA Librarian . This section discusses the benefits and drawbacks of each library type.

Benefits and Drawbacks of Library Types

The following information lists the benefits and drawbacks of each library type:

PDS

Benefits:

- Provides a familiar and standard means of storage for the source.
- No installation issues.
- Members can be read by other utilities such as compilers.

Drawbacks:

- Directory overhead becomes inefficient if there are more than 3-4K+ members.
- Strict maintenance is required to prevent x37 abends.

PDS/E

Benefits:

- Low maintenance.
- Current technology.
- Members can be read by other utilities such as compilers.

Drawbacks:

- Benchmarks show that the performance is slower than PDS or ELIB.
- Does not allow EXCP processing.

ELIB

Benefits:

- Low maintenance.
- Sophisticated set of utilities to manage the data sets.
- BDAM is more efficient for medium-sized directories, and VSAM is more efficient for large directories, than either PDS or PDS/E.
- Directory overhead routines are designed for a large number of members, thereby increasing efficiency.
- You can allocate a VSAM ELIB across multiple volumes. If you initially allocate an ELIB data set across two volumes, CA Endevor SCM will honor that allocation when ELIB expansion occurs. The expansion can be automatic when the page reserve limit is reached or explicitly performed through the use of BC1PNLIB. This is only for VSAM ELIB; BDAM ELIB must be allocated on a single volume.

Drawbacks:

- VSAM options can require a lot of processing time.
- Utilities are required for expands, copies, etc.
- Library members accessible only through CA Endevor SCM.

CA Panvalet and CA Librarian

Benefits:

- Accommodates the site standard.
- Directory compression issues are eliminated.
- Source output file can be read directly by CA Panvalet and CA Librarian utilities.

Drawbacks:

- With CA Panvalet, conflicts arise with CA Endevor SCM footprint and comment information.
- Library members are only accessible through CA Panvalet and/or CA Librarian.

Converting from One Library Type to Another

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

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

When a PDS load module is copied to a PDSE, CA Endevor SCM load module footprint information is retained only if the CA Endevor SCM processor utility BSTCOPY is used. Other utilities, such as IEBCOPY, do not have the capability to copy the *LOADMOD footprint. For more information, see TEC316937, How to Convert AllFusion Endevor Change Manager Load Libraries between PDS and PDSe Formats, on ca.com/support.

Using CA L-Serv for CA Endevor SCM's VSAM File Processing

CA L-Serv is a master started task that controls CA Endevor SCM VSAM files for Master Control Files (MCFs), packages, and ELIB (if you are using VSAM processing instead of BDAM processing). This task provides the following benefits:

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

Note: We recommend VSAM record level sharing (RLS) or CA L-Serv implementation to improve VSAM performance. For more information about RLS, see Using z/OS SYSPLEX VSAM Record Level Sharing (RLS) Support for MCFs and Package Data Set. For more information on how to implement CA L-Serv, see the *CA Common Services CA L-Serv Technical Bulletin* at http://ca.com/support.

Setting the RECBUFFSIZE Parameter

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

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

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

Monitoring CA L-Serv's Performance

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

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

Note: For more information about CA L-Serv's monitoring facilities, see the *CA Common Services CA L-Serv Technical Bulletin* at http://ca.com/support.

Evaluating Buffer Pool Usage

To see how well your buffer pools are being used, issue the DISPLAY BUFFERPOOL and DISPLAY STATISTICS SERVICE commands.

Note: For more information, see the *CA Common Services CA L-Serv Technical Bulletin* at http://ca.com/support.

Displaying Information about the Communications Server

The DISPLAY command provides additional options to provide online information about the Communication Server's activity.

Note: For more information, see the *CA Common Services CA L-Serv Technical Bulletin* at http://ca.com/support.

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

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

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

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

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

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

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

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

Sample JCL to enable RLS support may be found in iprfx.iqual.CSIQJCL, member BC1JDRLS.

VSAM Record Level Sharing provides the following performance enhancements:

- The VSAM buffers for ALL jobs and/or TSO users are consolidated into the SMSVSAM address space, increasing the chance of a record being in memory
- The SYSPLEX Lock Manager provides record level, CI level and CA level locking between SYSPLEX systems
- Due to the first two enhancements, CA Endevor SCM is able to bypass its own native Reserve/Release logic
- The I/O performance of the SMSVSAM address space and the SYSPLEX cache allows a significant reduction in the elapsed time required to do update I/Os.

Note: VSAM record level sharing (RLS) or CA L-Serv implementation is highly recommended.

Note: For more information about CA L-Serv, see Using CA L-Serv for CA Endevor SCM's VSAM File Processing.

Implementing RLS

CA Endevor SCM provides RLS support for Master Control Files (MCFs) and the Package dataset.

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

Tuning Your Processors

This section lists the points that you need to remember while tuning your processors.

Recommendations for Tuning Your Processors

When you are tuning your processors, consider the following to ensure your success:

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

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

- Avoid recursive executions of CA Endevor SCM by making use of the CONWRITE utility to output other elements. For example, if your jobcards are stored in one file and need to be merged into every executable file, CONWRITE can perform this merge without re-invoking CA Endevor SCM.

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

- Streamline processors by taking advantage of instream data (for example, DD *) and symbolics (for example, &C1SYSTEM). This eliminates extra steps that may have been required in the past.

- Allocate all temporary sequential data sets with BC1PDSIN to ensure that they are available for other programs such as CONLIST. Allocate other data sets using traditional JCL statements for each step.

- Ensure that your JCL dispositions are properly coded to release data sets when appropriate (for example, use FREE=CLOSE).

- Delete outputs with CONDELE wherever possible.

Tuning Your System

This section lists the points that you need to keep in mind while tuning your system.

Recommendations for Tuning Your System

When you are analyzing the general environment, consider the following to ensure your success:

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

- LRECL setting for variable block files:

  The output records from the compression routine CONCOMP3 can be larger than the input records when the input files contain binary data uploaded from other machines, or already compressed data (output from TRSMAIN).

  While it is not possible to predict the maximum increase in size of CONCOMP3 records, we have never seen a record increased by more than 1.6 times the size of the input record. Because the size of the output record depends on the input data, we can make predictions for specific input records, and suggest that you use the following recommendations for setting variable block files.

  - Set the block size (BLKSIZE) for variable block files to 27998, which is half a 3390 DASD track. This size allows very rapid I/O time and very efficient space usage. The reason for this is that MVS will put as many records as possible in the buffer defined by the BLKSIZEs and will only write the block if the remaining space cannot hold the next record. If a BLKSIZE of 2550 is chosen, then it will hold a maximum of 2546 bytes; however small BLKSIZEs are inefficient for disk use and also inefficient in I/O time.
  - The logical record lengths (LRECLs) for variable block files should be at least 4 less than 27998 (that is, 27994). MVS will use what it needs and as long as the buffer allocation (LRECL'value) is big enough, it will fit without problems or wasting space. There is no reason to reduce the LRECL to 255, for example. For variable or variable block files, the actual LRECL is not important. The LRECL does not define the length of the records. The actual size is defined in the records themselves. The LRECL defines the maximum length of any one record. As a consequence, setting a larger LRECL and BLKSIZE for variable block files is not a problem. As long as the LRECL is larger than the actual record, the record will fit.
  - For types that will contain binary data, we recommend a VB LRECL of 2048 or less, because this make source comparison perform more efficiently. The maximum LRECL is 13K. This allows for a compression result twice as large as the input.

Chapter 16: Altering Element Record Metadata

This section contains the following topics:

How to Alter Master Control File Element Metadata

Important: The Alter action changes Element metadata in the Master Control File and some changes cannot be easily reversed. Before you submit an Alter statement, we recommend that you back up the Master Control Files. If you wanted to undo changes, it might be easier to restore the Master Control Files. Otherwise, to undo changes that were made using name-masked values, you would probably need to code multiple Alter statements.

You can use the Alter action to change Element master record information (Element metadata) in the Master Control File. You specify an Element and what changes you want made to its metadata. Different metadata fields and multiple Element instances can be changed by the same Alter action. The Element metadata is changed at the location where the matching elements are found; the Alter action never fetches the Element.

The Alter action can be executed in Update or Noupdate mode. Noupdate mode produces the same output messages as Update mode; however the Master Control File Element record is not updated. Noupdate mode is an analysis tool that lets you see what effect the Update mode would have before you run it.

The Alter action is an Element action and is processed similarly to other Element actions, for example, the Add, Delete, Generate actions. However, the Alter action does not execute processors.

Using the Alter action, you can change the following Master Control File Element record metadata fields:

- Generate CCID— The CCID for the last Add, Update, Generate, Move, or Transfer action that executed the Generate processor against the Element.
- Last action CCID— The CCID for the last action that modified the Element.

- Retrieve CCID— The CCID for the last Retrieve action that was executed against the Element.
- Processor Group name— The current Processor Group for the Element.
- Signout UserID— The current user ID to which the Element is signed out.
- Description— The base description for the Element. This value is usually set by the Comment field when the Element is added.
- User Data— An 80-byte field stored with the Element. This value can also be set using a User Exit 2 (before action) program on an Add, Update, or Generate action.

To change the Master Control File element metadata, code the Alter statement. Submit the Alter statement for batch processing or include it in a Package. Packages can be submitted in foreground or batch. The Alter action is not available from the Foreground Options menu. Alter statements can be submitted from the Batch Options menu, but the SCL Generation menu does not have an option for the Alter action.

Note: For more information about the SCL syntax for the Alter statement, see Alter Statement in the *SCL Reference Guide*.

Alter Statements in Packages

You can include Alter statements in a Package. Packages can be submitted in foreground or batch, including those packages that contain Alter statements. However, the following limitations apply:

- When you create or modify a Package, you cannot use the Build Package Actions option on the Create/Modify Package panel to build the Alter action. The Build Package Actions option does not support building the Alter action.
- When submitted in a Package, valid Alter statement syntax is restricted as follows:
  - Wildcarding or masking of the from location fields (Environment, System, and so on) is not permitted.
  - The Alter action Search option is not permitted in a Package and if coded, the Package cast will fail.
- The default settings in the Optional Features table that affect package processing can be adjusted depending on the preferences of the administrator. For more information about the Optional Features table, see How to Set Up the Alter Action (see page 259).

Alter Action Security

Functional security settings, Element signout metadata, and the locking of Elements during Alter action processing control who can update which Elements. Alter action processing is secured by the following controls:

- Functional security— The Alter Action uses CA Endevor SCM security to verify that a user is authorized to perform the requested actions against the Element. The following security checks are performed:
  - Does the user have retrieve authority for the Element at the inventory location at which it is found? This check is performed before altering an Element.
  - Does the user have Override Signout authority at the inventory location at which the Element is found? This check is performed when the Override Signout clause is specified and the Element is not signed out to the current user.
  - If using ESI, does the user have Alter authority for the Element at the target Stage and Environment?
  - If using Endevor Native Security, does the user have Update authority for the Element at Stage 1 or the Entry Stage of the target Environment and Move authority for the Element at Stage 2?

  For information on how to setup security, see How to Set Up the Alter Action (see page 259).
- Signout UserID— The Alter Action does not alter Elements that are signed out to users other than the current user, unless the Override Signout option is specified.
  - To perform the Alter action when the Element is signed out to someone else, Override Signout must be used and the corresponding authority granted.
  - The Alter action retains the current Signout Userid for the Element; it does not update the Signout UserID with the user ID of the person performing the Alter action. If you want the Signout Userid Element field to be changed as part of the Alter action, use the Replace Signout UserID clause.
- Element locking— Serializing the Element applies to Update mode and Noupdate mode. The Alter action puts a lock on an Element when it is being processed. The following search criteria determine which Elements get locked:
  - When the Search option is specified— At the first Environment where the Element is found in the logical map when Search First option is used, or at all Environments where the Element is found in the logical map when the Search All option is used.
  - When the Nosearch option is specified— At the Environment specified, or if name-masking was used, at all the Environments that match the specified request.

Alter Action Processing

Alter action processing has the following conditions and results:

- Alter actions are processed in Global Type Sequence order and follow the same Type Sequence rules as other Element actions.
- Concurrent Action Processing supports the Alter action. If Concurrent Action Processing is enabled and requested, Alter actions are processed concurrently in Global Type Sequence order.
- The Alter action does not execute processors.
- Whenever an Element metadata field is updated, an SMF record is written for the updated field. The Element metadata field is updated and the SMF record is written, even if the Element metadata field value is identical before and after the update.
- The Alter Action invokes the following CA Endevor SCM User Exits: User Exit 1 (security), User Exit 2 (before action), and User Exit 3 (after action).
  - A User Exit 2 or User Exit 3 program can query all the Alter fields that are defined in the assembler $ALTRDS User Exit block or the COBOL ALTER-ACTION-BLOCK of EXITBLKS, but none of these fields can be modified.
  - A User Exit 2 program can modify the action CCID, the action Comment, and the Override Signout (Y/N) options located in the assembler $REQPDS block or REQUEST-INFO-BLOCK of the COBOL Exit block EXITBLKS.

    The signout check is performed on the Element at the target location after the User Exit 2 call, but before matching and Alter action processing are performed.
  - The Signout Userid and User Data Exit Request block fields cannot be modified for the Alter action.
  - The SMF record is not passed to the User Exit 3 program for the Alter action.
  - The Alter action exit control block contains information associated with the Alter action. This block is pointed to by the primary information exit control block. This block is only available at the time of User Exit 2 and User Exit 3 calls for the Alter action.

  Note: For more information about exits and the Alter action, see the *Exits Guide*.

How to Set Up the Alter Action

To enable the Alter action, the CA Endevor SCM administrator must ensure that the prerequisites are complete for functional security, Global Type Sequencing, and SMF recording. Optionally, the administrator can adjust the default settings in the Optional Features table that affect package processing depending on how they want the Alter action to be used in packages.

- **Review Alter action security**— The CA Endevor SCM administrator works with the site security administrator to enable certain user IDs to perform the Alter action.
  - The CA Endevor SCM administrator reviews the BC1TNEQU table to determine if the delivered setting is appropriate for their site. Permission to process the Alter action is determined by the security authorization function (SAF) set in the Endevor Security Interface (ESI) BC1TNEQU table. The following entry for the Alter action is included in the BC1TNEQU sample that is included with the Version 17.0 release: FUNCEQU SAFAUTH=ALTER,C1ACTNS=(ALTER). As delivered, the SAF authorization value of Alter is equated with the Alter access level (c1access). This entry restricts access to the Alter action to users who have ALTER authority for the ESI pseudo data set name.

    Important: If you already have a BC1TNEQU table in place, you must add a statement for the Alter action to control its use; otherwise, *anyone can execute the Alter action*.
  - The site security administrator ensures that the appropriate user IDs are associated to the Alter access level.

  Note: For more information about security, see How to Enable Functional Security in the *Security Guide*.

- **Enable Global Type Sequencing**— Global Type Sequencing may already be enabled at your site, because it is a prerequisite for other functions. Global Type Sequencing determines the order in which Types are processed in batch.

  For Global Type Sequencing to be in effect, both the Parmlib data set and the Type Sequence member must be defined to the Defaults table C1DEFLTS.

  Note: For more information about Global Type Sequencing, see How to Enable Global Type Sequencing in the *Scenario Guide*.

- **Enable the SMF option**— The SMF option may already be enabled at your site, because it is a prerequisite for other functions. The SMF option can record each action and each security violation that occurs during CA Endevor SCM processing and is set per Environment.

  For the SMF option to be in effect, the Defaults table (C1DEFLTS) must include the following parameters:

  - The SMFREC# parameter must be set in the TYPE=MAIN portion of the Defaults table. SMFREC# sets the SMF record number that is assigned to SMF records written by CA Endevor SCM at this site.
  - The parameter SMFACT=Y must be set in the TYPE=ENVIRONMENT portion of the Defaults table to write out SMF action records.

  If you want to track security violations for your own auditing purposes, make sure that SMFSEC=Y is set in the TYPE=ENVIRONMENT portion of the Defaults table to write out SMF security records.

  Note: For more information about the SMF option, see Enable SMF Recording in the *Administration Guide*.

- **(Optional) Configure Package Processing Options—** The Optional Features table includes the following options that affect package processing. Depending on how you want the Alter action to be used, adjust the default settings as appropriate:

  - **Limit processing to one action in the same package**— (Optional) You can limit package processing to one action against the same Element instance within the same package. This option disallows additional actions that modify the same Element instance within the same package. This restriction applies to any action, not only to Alter actions. The following message is delivered with the severity level of C (for caution with a return code of 8):

    ```
    C1G0507C  ELEMENT IS REFERENCED BY MULTIPLE ACTIONS WITHIN THIS PACKAGE
    ```

    To disallow more actions, set the severity level for this message to E (for error with a return code of 12) using this parameter in update the Optional features table, ENCOPTBL:

    ```
    ENHOPT MSGSEVERITY_C1G0507=C              W/C/E     0002833
    ```

  - **Allow nonpackage actions—** (Optional) If you want to use the Alter action outside of a package against an Element in an inventory location that usually requires a package, add the Alter action to this option:

    ALLOW_NON_PKG_ACTIONS— Specifies actions that can be performed outside of a package, even if the inventory location would usually require a package.

  - **Allow packages containing Alter actions only to skip approval process—** (Optional) If you want to use the Alter action in a package, but want to skip the package approval process, add the Alter action to this option:

    PKG_ACTIONS_NO_APPRVR— Specifies actions in a package that do not require approval.

Review Alter Action Activity

To view or verify the effects of an Alter action on an Element, you have the following options:

- Review the Alter action messages and return codes (see page 262).
- View the Element master panels (see page 263) for the following information:
  - Current metadata values for the Element, which may have been changed by Alter actions, including the User Data field.
  - Metadata about Alter actions that were performed against the Element.
- Print the Element master to view the current metadata, which may have been changed by Alter actions. Currently, printing the Element master does not print the User Data field or the metadata about the Alter actions that were performed against the Element. For more information about how to print, see Print Element Information in the *User Guide*.
- Review the Alter action metadata fields returned by the List Element function. You can use the following options to access this data:
  - An API program. For more Information about the Alter action metadata response structure fields for the API function List Element, see Information about Alter Action Metadata in the chapter Using API Function Calls in the *API Reference Guide*.
  - The Comma Separated utility. For more information about the Alter action metadata fields extracted by the List Element option of the Comma Separated utility, see Alter Action Metadata, in the chapter "Using the Comma Separated Value (CSV) Utility" in the *Utilities Guide*.
- Run and view Endevor SMF reports. (see page 266)
- List SMF data using an API program. (see page 267)

Alter Action Messages and Return Codes

You can review the messages and return codes issued during Alter action processing to verify that the intended action was performed.

By default the Action Execution report and the Action Summary report are written to the C1MSGS1 DD statement. If your batch execution JCL is coded with the C1MSGS1 and C1MSGS2 DD statements the Action Execution Report is written to the C1MSGS1 file and the Execution Summary report is written to the C1MSGS2 file. The execution report can be viewed online by browsing data sets userid.C1TEMPR1.MSGS or userid.C1TEMPR2.MSGS.

Various messages are issued during Alter action processing, including the following:

- If the from-value is omitted or fully wildcarded, the mask-value in message **C1G0076I** is set to an asterisk:
  `C1G0076I  MASK: FIELD-NAME MASK-VALUE`
- The Alter action changes the Master Control File for each matching Element at the location where the Element resides. No copyback logic occurs. Message C1G0280I prints the location where the Element was located and updated:
  `C1G0280I  ELEMENT AT LOCATION environment/stage id/system name/subsystem name/type name SELECTED FOR PROCESSING`
- C1MSGS2 Summary Report— Return codes indicate whether all, some, or no fields matched the mask. *NOUPDT* denotes Noupdate mode in which no updates were performed.

Return codes are associated with certain conditions. In the following descriptions of the return codes, the Replace clause matches when the Alter action statement specifies a value for the Master Control File field you want to replace and the specified value matches what is currently in the Master Control File field.

Note: When Noupdate is active, report messages and return codes simulate what would happen in Update mode.

RC=0

Indicates that the Alter request completed 100% successfully. All requested Master Control File fields are (or will be) updated as specified by the submitted Alter action syntax. RC=0 is returned in the following cases:

- All Replace clauses match and Update option is specified, therefore the Master Control File record is updated.
- All Replace clauses match but Noupdate option is specified, therefore the Master Control File record is **not** updated.
- No from-value is provided and Update option is specified. The Master Control File record is updated.
- No from-value is provided and Noupdate option is specified. The Master Control File record is **not** updated.

RC=4

Indicates that the Alter request only partially completed. One or more of the request fields in the SCL are not (or will not be) updated. RC=4 helps you to determine that some Master Control File fields were updated successfully, but not all, without having to read all report messages. RC=4 is returned in the following cases:

- Some Replace clauses match and Update option is specified, therefore only the matched Master Control File fields are updated, while the non-matched fields are bypassed.
- Some Replace clauses match but Noupdate option is specified, therefore the Master Control File record is **not** updated.
- No Elements found.

RC=8

Indicates that no from-values matched the current Master Control File fields for any of the replace clauses specified. RC=8 lets you quickly determine which Elements were not updated at all; identify possible error conditions or an invalid SCL request. RC=8 is returned in the following cases:

- No Replace clauses match and Update option is specified, therefore the Master Control File record is **not** updated.
- No Replace clauses match and Noupdate option is specified, therefore the Master Control File record is **not** updated.

RC=12

Indicates invalid syntax or specification.

Note: For more information about messages, see the *Messages and Codes Reference Guide*.

Element Master Information About Alter Actions

The Element Master panels display metadata values for a particular Element. The following types of information are displayed:

- **Current metadata values for the Element**— View the current metadata for the Element to verify that the metadata values are as you expected.
- **Information about Alter actions that were performed against the Element (Alter action metadata)**— The Alter Action section of the Element record in the Master Control File provides information about the Alter actions that were run against the Element. This metadata concerns the Alter action itself, for example the last date and time that the Alter action was run against the Element. You can see this Alter action metadata displayed on the last panel of the Element Master panel in the Alter Action section.

Navigate to the end of the last Element Master panel to find the Alter Action section. Review the following fields in the Alter Action section to see information about the Alter actions that were run against this Element:

- **First Date/Time—** Indicates the date and time that the first Alter action was performed against this copy of the Element.
- **Cumulative # of Fields Altered—** Indicates the total number of times that fields were updated by all the Alter actions performed against this copy of the Element.
- **Last Date/Time—** Indicates the date and time that the last Alter action was performed against this copy of the Element. If the First Date/Time matches the Last Date/Time, then the Alter action was only run once.
- **Last UserID—** Indicates the user ID associated with the last Alter action performed against this copy of the Element.
- **Fields Updated by the Last Alter Action—** Indicates whether any of the following Master Control File Element record metadata fields were updated for the last date and time the Alter action was run against this Element: Generate CCID, Last Act CCID, Retrieve CCID, Description, Processor Group, Signout UserID, and User Data. Valid values are Y for yes or N for no.

  These fields are reset after every Alter action. For example, if an Alter action replaced the Generate CCID and the Last Action CCID, then those field values would be Y and the other values would be N. If the next Alter action replaced the Processor Group, then only the Processor Group would be Y and the other fields, including the Generate CCID and Last Action CCID, would be N.

Note: The Last Element Action section on the Element Master panels is not changed by the Alter action, or by Signin or Retreive.

Note: Certain Element actions clear the Alter action metadata fields. For more information, see Element Actions that Clear Alter Action Metadata (see page 264) and Element Actions that Retain Alter Action Metadata (see page 265).

Element Actions that Clear Alter Action Metadata

When actions are performed against an Element, the Alter action metadata fields are retained or cleared depending on the action. The Alter action metadata fields are **cleared** when the following actions are performed against an Element:

- Add or Update that causes a fetch to the target location— For example: Suppose you add an existing Element from a production Environment to a target location lower in the map. And suppose that the existing Element had been altered and had Alter metadata before it was fetched back to the target. Then the Element that was added at the target location, does not include the Alter action metadata.

- Generate with Copyback or Nosource— For example:
  - If you Generate Element Y at Stage 2 with Copyback to Stage 1, then the Alter metadata gets cleared at Stage 1.
  - If you Generate Element Y at Stage 2 with Nosource to Stage 1, then the Alter metadata gets cleared at Stage 1.
- If your site uses the *CA Endevor SCM Quick Edit* option, when Edit is used to fetch an element back to a lower stage.

Element Actions that Retain Alter Action Metadata

When actions are performed against an Element, the Alter action metadata fields are retained or cleared depending on the action. The Alter action metadata fields are **retained** when the following actions performed against an Element:

- Move— For example: If you Move Element Y from Stage 2 with or without history, to the next location in the map, then the Alter metadata is retained at the target location. If the Bypass Element Delete option is coded on the Move action, the Alter metadata is retained at the source location.
- Transfer— For example: If you Transfer Element Y from Stage 2 to an archive file, then Transfer it back from the archive file, the Alter metadata is retained at the target location. This is true even if the target location is a different location than where it was archived from.
- Restore
- Generate in place
- Update in place— For example: If you Update Element Y that resides at Stage 1, then the Alter metadata is retained at Stage 1.
- Archive
- Copy
- Delete only components
- Signin

SMF Recording

You can audit Alter action activity by reviewing SMF records. SMF recording, which can record security violations and action activity, must be active to enable the Alter action.

- Security records can show if a user attempted to perform an Alter action for which they do not have permission. For example, suppose someone attempted to alter an element at an Environment they did not have access to, an error would be produced. The security record would log who attempted to perform the failed action.
- Activity records show if the Alter action was performed and what effect the action had on the Element record in the Master Control File.

Endevor SMF Reports

SMF historical reports summarize security violations and Element activity recorded by CA Endevor SCM. These reports are available when SMF logging is in use at your site and are generated using the SMF records written during CA Endevor SCM processing. The following reports show Alter action activity:

CONRPT40— Security Violation Summary. For each system requested, this report gives a detailed account of each security violation that occurred. Specifically, this report lists each attempt—by any user—to perform an unauthorized action. Attempts to perform unauthorized Alter actions are recorded.

CONRPT42— Element Activity Profile. This report details each action performed against the Elements within a particular system, subsystem, Element type, and stage. Using this report, you can determine exactly which Elements were changed using the Alter action. Also, shown are the Replace clause specified on the Alter action and the Master Control File Element record value before and after it was replaced and the mask value that was specified on the Alter action. If the Replace User Data clause was specified, the additional parameters are shown.

Note: For more information about the reports, see the chapter "Historical (SMF) Reports" in the *Reports Guide*.

API List SMF Function for Alter Action

The API List SMF function lets you list the SMF record data corresponding to activity related to CA Endevor SCM actions. The ALSMFA_RS Response Structure includes Alter action fields that indicate the Master Control File Element record field to which the Alter action requested a change and whether the field was changed. The Alter action data area contains the before, after, and mask values for the Master Control File Element record field that was processed.

Note: For more information about Alter action fields for the List SMF function, see ALSMFA_RS Response Structure Fields for Activity Requests in the *API Reference Guide*.

Appendix A: Converting Delta Formats

This section contains the following topics:

How to Convert Forward Delta Formats to Reverse Delta Formats

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

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

Considerations When Converting Forward Delta Formats to Reverse Delta Formats

Consider the following when converting to reverse delta format to ensure your success:

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

Analyzing Existing Types

Use reverse deltas for the types that:

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

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

- Copybooks
- JCL
- Source

Use forward deltas for types that:

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

Resizing and Allocating New Base Libraries

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

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

Resizing the Delta Libraries

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

Evaluating and Modifying Processors

Evaluate your processors, keeping in mind the following:

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

Running a Full Unload of Each Environment

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

Adjusting Type Definitions

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

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

Reloading Inventory by System

Execute the Reload utility (BC1JRELD) by system to populate the new libraries with your inventory.

Validating the Results

Execute the Validate utility (BC1JVALD) to confirm the results.

How to Convert a Forward/Reverse Delta to a Full-Image Delta

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

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

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

Source Format	Target Format	MOVE with history	MOVE without history	TRANSFER with history	TRANSFER without history
Full-image	Full-image	Yes	Yes	Yes	Yes
Full-image	Reverse	No	Yes	No	Yes

Source Format	Target Format	MOVE with history	MOVE without history	TRANSFER with history	TRANSFER without history
Full-image	Forward	No	Yes	No	Yes
Reverse	Full-image	No	Yes	No	Yes
Forward	Full-image	No	Yes	No	Yes

Restore Action and Converting Delta Formats

The Restore statement restores an element from an archive data set back to CA Endevor SCM, copying the source as it was before the element was archived or transferred to the data set. Also, the Restore action can restore elements from an unload CA Endevor SCM file.

You can use the Restore action to convert elements to different storage formats as follows:

- Reverse delta elements to forward delta elements
- Forward delta elements to reverse delta elements
- Forward, reverse, or image delta elements to log delta elements

Note: For more information, see Restore Syntax in the *SCL Reference Guide*.

How to Convert Elements to Log Delta Format

You can use the Restore action to convert forward, reverse, or image delta elements to log delta elements. Complete the following steps to convert to log delta format:

1. Use the Transfer to Archive action with the Bypass Delete Processor option to archive the elements you want to change.

   This archives the element and component base and delta files in their current formats, deletes them from CA Endevor SCM, but leaves the elements outputs.

2. Change the Type definition to Log on the elements in their current location.

3. Use the Restore action with the Bypass Generate Processor to restore the elements from the archive to the element's current location.

   The element delta type is changed to log format and the outputs are not changed.

   Note: If you want to retain the MCF generate information, then *also* specify the Retain Generate History option. Without the Retain Generate History option, the Restore action reinitializes the generate MCF information and the prior information is lost.

Appendix B: Using Catalog Utilities

This section contains the following topics:

How to Build the Element Catalog

CA Endevor SCM uses an Element Catalog file to support long element names and to boost performance by reducing the volume of I/O operations. When upgrading from release 3.9 or earlier, you perform several post-installation steps. These steps are summarized in the following list and described in detail in the following sections.

1. Copy all your MCF VSAM data sets to new data sets with VSAM attributes with CSIQJCL member/JOB BC1JXMCF.
2. Copy your Package VSAM data set to a new data set with VSAM attributes with CSIQJCL member/JOB BC1JXPCF.
3. Define your new catalog VSAM data set with CSIQJCL member/JOB BC1JJB07.
4. Add the ELMCATL= catalog parameter to the C1DEFLTS table in *iprfx.iqual.CSIQSRC* and assemble and link-edit C1DEFLTS.
5. Run CA Endevor SCM's Catalog Build utility against all defined Environments with CSIQJCL member/JOB BC1JXCNV.

Convert Your Existing MCF VSAM Data Sets

The MCF data set's maximum record length has changed from release 3.9 and earlier. As a result, you must redefine all your stage MCFs for all your environments. Use member BC1JXMCF from your CA Endevor SCM JCL Library to tailor the job stream for all your MCF file conversions. This job backs up your existing data sets to sequential files, deletes and redefines the MCF VSAM files, then populates the records back into your newly-defined MCF file clusters. This job is set up to handle a single environment. You need to modify it to include all the environments defined at your site.

Convert Your Existing Package VSAM Data Sets

The Package data set's maximum record length has also changed from release 3.9 and earlier. Use member BC1JXPCF from your CA Endevor SCM JCL library to tailor the job stream to do your package file conversion. This job backs up your existing data set to a sequential file, deletes and redefines the package file, populates the newly-defined VSAM package file with your package data.

Define the MCF Catalog Data Set

Beginning with Release 4.0, a catalog data set is used to keep track of elements across all defined environments in your C1DEFLTS. Use member BC1JJB07 from your CA Endevor SCM JCL library to define the catalog data set. You will need to tailor the job stream based upon your installation's needs.

Important! Once you convert to Release 4.0 or above, you must not use a 3.9 or earlier release to access Release 4.0 or above data. If you do, the catalog becomes out of sync with the MCFs.

Update the C1DEFLTS Table

Update your C1DEFLTS table to identify your element catalog data set to CA Endevor SCM by using the ELMCATL parameter. Remember to compile and link it into your site's proper authorized library.

```
C1DEFLTS TYPE=MAIN,                    X
     ELMCATL='CA.PROD.ELMCATL',         X
```

Run the Catalog Build Utility

Run the catalog utility to populate the catalog data set with element information. This utility should only be run during the conversion process and it must be run against all environments defined in your C1DEFLTS table.

BC1PXMCS allows you to selectively choose the environments you want to populate. You can run it against each environment separately or against all environments collectively in a single step. Before you begin to use CA Endevor SCM, you must have run the conversion utility against all environments defined in your C1DEFLTS table. Use member BC1JXCNV from your CA Endevor SCM JCL library to tailor the job stream to do your initial build of the element catalog.

Note: The person running this job must have the authority to access all the environments in the C1DEFLTS table (if BSTIPT is omitted) or all environments specified on the BSTIPT statement. Otherwise, the catalog contents will be incomplete and may result in "element not found" errors.

The output report of running the BC1JXCNV job is written to BSTLST. This report contains the number of catalog segments read and written to the catalog data set.

Sample Output Report

This section describes the sample output of running the BC1JXCNV utility that loaded an empty catalog data set from a C1DEFLTS table, containing two environments, TEST and PROD, with BSTIPT omitted. The output generated by BC1PXMCS, BC1PXCCS, and BC1PXCSC to BSTLST is described next.

BC1PXMCS in step 3 creates element segment records from MCFS and generates the following report:

```
CA Endevor SCM MCF Conversion Element Extract UTILTIY LOG
MCF CATALOG SEGMENT DATA WILL BE CREATED FOR ENVIRONMENT TEST
NUMBER OF CATALOG SEGMENTS WRITTEN FOR ENVIRONMENT TEST: 9243
MCF CATALOG SEGMENT DATA WILL BE CREATED FOR ENVIRONMENT PROD
NUMBER OF CATALOG SEGMENTS WRITTEN FOR ENVIRONMENT PROD: 5978
TOTAL NUMBER OF CATALOG SEGMENTS WRITTEN, ALL ENVIRONMENTS: 15221
```

BC1PXCCS in step 4 extracts segment records and produces the following report:

```
CA Endevor SCM Catalog Segment Extraction UTILITY LOG
CCS0008I TOTAL NUMBER OF CATALOG SEGMENTS READ: 0
CCS0009I TOTAL NUMBER OF CATALOG SEGMENTS WRITTEN: 0
```

If you are loading an empty catalog data set, the number of catalog segments read and written should be zero.

BC1PXCSC in step 6 creates catalog records from the catalog segments and generates the following output:

```
CA Endevor SCM MCF Element Segment Conversion to Catalog Format UTILITY LOG
TOTAL NUMBER OF ELEMENT ENTRIES READ, ALL ENVIRONMENTS: 15221
TOTAL NUMBER OF CATALOG ENTRIES WRITTEN, ALL ENVIRONMENTS: 13366
TOTAL NUMBER OF ELEMENT INDEX RECORDS WRITTEN, ALL ENVIRONMENTS: 10183
```

You can also use the BC1JXCNV utility to add a new, populated environment to an existing C1DEFLTS table and catalog data set. In the following sample output, the ARCHIVE environment is added to an existing catalog data set. The output reports produced by step 3, step 4 and step 6 are described next.

BC1PXMCS in step 3 generates the following output:

```
CA Endevor SCM MCF Conversion Element Extract UTILTIY LOG
ENV ARCHIVE
MCF CATALOG SEGMENT DATA WILL BE CREATED FOR ENVIRONMENT ARCHIVE: 75
TOTAL NUMBER OF CATALOG SEGMENTS WRITTEN, ALL ENVIRONMENTS: 75
```

BC1PXCCS in step 4 generates the following output:

```
CA Endevor SCM Catalog Segment Extraction UTILITY LOG
NUMBER OF CATALOG SEGMENTS WRITTEN: 2500
NUMBER OF CATALOG SEGMENTS WRITTEN: 5000
NUMBER OF CATALOG SEGMENTS WRITTEN: 7500
NUMBER OF CATALOG SEGMENTS WRITTEN: 10000
NUMBER OF CATALOG SEGMENTS WRITTEN: 12500
NUMBER OF CATALOG SEGMENTS WRITTEN: 15000
TOTAL NUMBER OF CATALOG SEGMENTS READ: 15221
TOTAL NUMBER OF CATALOG SEGMENTS WRITTEN: 15221
```

BC1PXCSC in step 6 generates the following output:

```
CA Endevor SCM MCF Element Segment Conversion to Catalog Format UTILITY LOG
TOTAL NUMBER OF ELEMENT ENTRIES READ, ALL ENVIRONMENTS: 15296
TOTAL NUMBER OF CATALOG ENTRIES WRITTEN, ALL ENVIRONMENTS: 13420
TOTAL NUMBER OF ELEMENT INDEX RECORDS WRITTEN, ALL ENVIRONMENTS: 10215
```

The Catalog Rename Utility

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

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

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

The utility can run in following two modes:

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

To execute the rename utility, use member BC1JXCNM from your JCL library. To invoke validate or update mode, change the PARM= parameter on the execute statement.

The Catalog Synchronization Utility

The CA Endevor SCM Catalog Synchronization Utility (BC1PCSYN) allows you to update the element catalog to recover differences between the element catalog and its related MCF files.

This utility has two modes, a validate mode and an update mode. The validate mode simply reports on the differences between the catalog and MCF files. The update mode actually recovers the differences between the element catalog and its related MCFs by updating the element catalog file.

Under the update mode, the utility synchronizes the element catalog and the MCF element records in two phases.

The first phase validates MCF records against the catalog. A catalog segment is created for any MCF record found missing from the catalog. A catalog segment is updated with MCF element data when the associated catalog data is found to be different.

The second phase validates the catalog to MCF in order to remove dead segments from the catalog. All segments which do not have a MCF element record are removed from the catalog.

Under validate mode, the utility performs the same two-phase check but does not modify the catalog. Any differences are simply reported in the execution log.

To select the mode, use the PARM= parameter on the JCL execution statement. 4.

In both phases, processing is done by environment. You can specify which environments the utility will process, or allow the environment selection to default to all environments defined in the C1DEFLTS table.

Important! Do not use this utility to initially load the catalog file. Use job BC1JXCNV to load the catalog file. For more information, see Run the Catalog Build Utility (see page 277).

Catalog Synchronization Utility JCL

To execute the synchronization utility (BC1PCSYN), use JCL member BC1JCSYN, which is available in your site's CA Endevor SCM CSIQJCL data set. Edit this member to conform to the needs of your installation before executing the job.

Catalog Synchronization Utility Syntax

To select which environments the utility will process, you need to edit the BSTIPT DD JCL statement. If the DD statement is omitted or if the file is empty (no input syntax), all environments in the C1DEFLTS table are selected. The BSTIPT file is a 80 character fixed record file. The syntax follows the same parsing rules as CA Endevor SCM SCL statements.

To select a single environment, specify the following:

```
ENVIRONMENT env-name.
```

To select multiple environments on a single line, specify the following:

```
ENVIRONMENT (env-name, env-name, env-name).
```

To select multiple environments on multiple lines, specify the following:

```
ENVIRONMENT (env-name, env-name, env-name,
  env-name, env-name).
```

Catalog Synchronization Utility Sample Reports

The following sample report results from executing the utility in validate mode against a test CA Endevor SCM system with three environments. In this example, the first two environments are out of sync with its catalog.

```
1  (C) 2005 Computer Associates International, Inc.                                    10NOV04 14:33:53       PAGE   1

                                      Endevor MCF Catalog Synchronization UTILITY LOG             RELEASE 7.0  SERIAL B7000C




 SYN0000I  BEGINNING PHASE 1



 SYN0002I  ENVIRONMENT ENV1     MCF ELEMENT TO CATALOG SYNCRONIZATION STARTING


 *MISSING* CTLG SGMT: ELEMENT EJZZBLONGNAME02

                TYPE JDOHFS   AT ENV ENV1     SID 1 SYS SYST100  SBS SBS100

 *MISSING* CTLG SGMT: ELEMENT JBEANlongld5VERYLONGNAMEverylongnameVERYLONGNAMEverylongnameLAROSE
```

```
                TYPE JDOHFS   AT ENV ENV1     SID 1 SYS SYST100  SBS SBS100

*MISSING* CTLG SGMT: ELEMENT EJudylarose01

                TYPE JDOHFS   AT ENV ENV1     SID 1 SYS SYST100  SBS SBS100

*OLD    * CTLG SGMT: ELEMENT EJudylarose02

                TYPE JDOHFS   AT ENV ENV1     SID 1 SYS SYST100  SBS SBS100

*OLD    * CTLG SGMT: ELEMENT IA6PGM

                TYPE ASMPGM   AT ENV ENV1     SID 1 SYS SYST100  SBS SBS100

*OLD    * CTLG SGMT: ELEMENT OLENIC01

                TYPE JDOHFS   AT ENV ENV1     SID 2 SYS SYST100  SBS SBS100


SYN0003I  TOTAL # OF MISSING CTLG SGMTS FROM ENVIRON ENV1    :        3

SYN0004I  TOTAL # OF OLD CTLG SGMTS FOUND IN ENVIRON ENV1    :        3

SYN0005I  TOTAL # OF ELEMENTS PROCESSED FROM ENVIRON ENV1    :      102


SYN0002I  ENVIRONMENT ENVA     MCF ELEMENT TO CATALOG SYNCRONIZATION STARTING


*MISSING* CTLG SGMT: ELEMENT TESTJO1

                TYPE JCLI     AT ENV ENVA     SID 2 SYS SYSTA00  SBS SBSA00

*MISSING* CTLG SGMT: ELEMENT SANELM1

                TYPE JCLI     AT ENV ENVA     SID 2 SYS SYSTC00  SBS SBSC00


SYN0003I  TOTAL # OF MISSING CTLG SGMTS FROM ENVIRON ENVA    :        2

SYN0004I  TOTAL # OF OLD CTLG SGMTS FOUND IN ENVIRON ENVA    :        0

SYN0005I  TOTAL # OF ELEMENTS PROCESSED FROM ENVIRON ENVA    :        5


SYN0002I  ENVIRONMENT ENVZ     MCF ELEMENT TO CATALOG SYNCRONIZATION STARTING



SYN0003I  TOTAL # OF MISSING CTLG SGMTS FROM ENVIRON ENVZ    :        0
```

```
 SYN0004I  TOTAL # OF OLD CTLG SGMTS FOUND IN ENVIRON ENVZ    :        0

 SYN0005I  TOTAL # OF ELEMENTS PROCESSED FROM ENVIRON ENVZ    :        1


 SYN0011I  TOTAL # OF ELEMENTS PROCESSED, ALL ENVIRONMENTS:          108




 SYN0000I  BEGINNING PHASE 2




 SYN0031I  ENVIRONMENT ENV1     CATALOG TO MCF ELEMENT SYNCRONIZATION STARTING.


 *ORPHAN * CTLG SGMT: ELEMENT EJudylarose01

                TYPE JDOHFS   AT ENV ENV1     SID 2 SYS SYST100  SBS SBS100

 *ORPHAN* CTLG SGMT: ELEMENT EJZZBLONGNAME02

1  (C) 2005 Computer Associates International, Inc.                                              10NOV04 14:34:03       PAGE   2

                                          Endevor MCF Catalog Synchronization UTILITY LOG              RELEASE 7.0  SERIAL B7000C




                TYPE JDOHFS   AT ENV ENV1     SID 2 SYS SYST100  SBS SBS100

 *ORPHAN * CTLG SGMT: ELEMENT JBEANlongld5VERYLONGNAMEverylongnameVERYLONGNAMEverylongnameLAROSE

                TYPE JDOHFS   AT ENV ENV1     SID 2 SYS SYST100  SBS SBS100

 SYN0032I  TOTAL # OF ORPHAN CTLG SGMTS     IN  ENVIRON ENV1    :        3

 SYN0033I  TOTAL CATALOG SEGMENTS PROCESSED FOR ENVIRON ENV1    :      105


 SYN0031I  ENVIRONMENT ENVA     CATALOG TO MCF ELEMENT SYNCRONIZATION STARTING.


 *ORPHAN * CTLG SGMT: ELEMENT SANELM1

              TYPE JCLI     AT ENV ENVA     SID 1 SYS SYSTC00  SBS SBSC00

 *ORPHAN * CTLG SGMT: ELEMENT TESTJO1
```

```
              TYPE JCLI     AT ENV ENVA     SID 1 SYS SYSTA00  SBS SBSA00

 SYN0032I  TOTAL # OF ORPHAN CTLG SGMTS     IN  ENVIRON ENVA    :        2

 SYN0033I  TOTAL CATALOG SEGMENTS PROCESSED FOR ENVIRON ENVA    :        7


 SYN0031I  ENVIRONMENT ENVZ     CATALOG TO MCF ELEMENT SYNCRONIZATION STARTING.


 SYN0032I  TOTAL # OF ORPHAN CTLG SGMTS     IN  ENVIRON ENVZ    :        0

 SYN0033I  TOTAL CATALOG SEGMENTS PROCESSED FOR ENVIRON ENVZ    :        1


 SYN0034I  TOTAL CATALOG SEGMENTS PROCESSED, ALL ENVIRONMENTS:         113
```

If the utility is executed again, but this time in update mode, the catalog will be repaired and the following log report will be produced.

```
1  (C) 2005 Computer Associates International, Inc.                                        10NOV04 14:33:53       PAGE   1

                                        Endevor MCF Catalog Synchronization UTILITY LOG             RELEASE 7.0  SERIAL B7000C




 SYN0000I  BEGINNING PHASE 1



 SYN0002I  ENVIRONMENT ENV1     MCF ELEMENT TO CATALOG SYNCRONIZATION STARTING


 *CREATED* CTLG SGMT: ELEMENT EJZZBLONGNAME02

               TYPE JDOHFS   AT ENV ENV1     SID 1 SYS SYST100  SBS SBS100
```

```
*CREATED* CTLG SGMT: ELEMENT JBEANlongld5VERYLONGNAMEverylongnameVERYLONGNAMEverylongnameLAROSE

                TYPE JDOHFS   AT ENV ENV1     SID 1 SYS SYST100  SBS SBS100

*CREATED* CTLG SGMT: ELEMENT EJudylarose01

                TYPE JDOHFS   AT ENV ENV1     SID 1 SYS SYST100  SBS SBS100

*UPDATED* CTLG SGMT: ELEMENT EJudylarose02

                TYPE JDOHFS   AT ENV ENV1     SID 1 SYS SYST100  SBS SBS100

*UPDATED* CTLG SGMT: ELEMENT IA6PGM

                TYPE ASMPGM   AT ENV ENV1     SID 1 SYS SYST100  SBS SBS100

*UPDATED* CTLG SGMT: ELEMENT OLENIC01

                TYPE JDOHFS   AT ENV ENV1     SID 2 SYS SYST100  SBS SBS100


SYN0003I  TOTAL CATALOG SEGMENTS CREATED FOR ENVIRON ENV1    :        3

SYN0004I  TOTAL CATALOG SEGMENTS UPDATED FOR ENVIRON ENV1    :        3

SYN0005I  TOTAL #  OF ELEMENTS PROCESSED FOR ENVIRON ENV1    :      102


SYN0002I  ENVIRONMENT ENVA     MCF ELEMENT TO CATALOG SYNCRONIZATION STARTING


*CREATED* CTLG SGMT: ELEMENT TESTJO1

                TYPE JCLI     AT ENV ENVA     SID 2 SYS SYSTA00  SBS SBSA00

*CREATED* CTLG SGMT: ELEMENT SANELM1

                TYPE JCLI     AT ENV ENVA     SID 2 SYS SYSTC00  SBS SBSC00


SYN0003I  TOTAL CATALOG SEGMENTS CREATED FOR ENVIRON ENVA    :        2

SYN0004I  TOTAL CATALOG SEGMENTS UPDATED FOR ENVIRON ENVA    :        0

SYN0005I  TOTAL #  OF ELEMENTS PROCESSED FOR ENVIRON ENVA    :        5


SYN0002I  ENVIRONMENT ENVZ     MCF ELEMENT TO CATALOG SYNCRONIZATION STARTING



SYN0003I  TOTAL CATALOG SEGMENTS CREATED FOR ENVIRON ENVZ    :        0
```

```
 SYN0004I  TOTAL CATALOG SEGMENTS UPDATED FOR ENVIRON ENVZ    :        0

 SYN0005I  TOTAL #  OF ELEMENTS PROCESSED FOR ENVIRON ENVZ    :        1


 SYN0011I  TOTAL # OF ELEMENTS PROCESSED, ALL ENVIRONMENTS:          108




 SYN0000I  BEGINNING PHASE 2




 SYN0031I  ENVIRONMENT ENV1     CATALOG TO MCF ELEMENT SYNCRONIZATION STARTING.


 *DELETED* CTLG SGMT: ELEMENT EJudylarose01

                TYPE JDOHFS   AT ENV ENV1     SID 2 SYS SYST100  SBS SBS100

 *DELETED* CTLG SGMT: ELEMENT EJZZBLONGNAME02

1  (C) 2005 Computer Associates International, Inc.                                                10NOV04 14:34:03       PAGE   2

                                          Endevor MCF Catalog Synchronization UTILITY LOG              RELEASE 7.0  SERIAL B7000C




                TYPE JDOHFS   AT ENV ENV1     SID 2 SYS SYST100  SBS SBS100

 *DELETED* CTLG SGMT: ELEMENT JBEANlongld5VERYLONGNAMEverylongnameVERYLONGNAMEverylongnameLAROSE

                TYPE JDOHFS   AT ENV ENV1     SID 2 SYS SYST100  SBS SBS100

 SYN0032I  TOTAL CATALOG SEGMENTS DELETED   FOR ENVIRON ENV1    :        3

 SYN0033I  TOTAL CATALOG SEGMENTS PROCESSED FOR ENVIRON ENV1    :      105


 SYN0031I  ENVIRONMENT ENVA     CATALOG TO MCF ELEMENT SYNCRONIZATION STARTING.


 *DELETED* CTLG SGMT: ELEMENT SANELM1

              TYPE JCLI     AT ENV ENVA     SID 1 SYS SYSTC00  SBS SBSC00

 *DELETED* CTLG SGMT: ELEMENT TESTJO1
```

```
              TYPE JCLI     AT ENV ENVA     SID 1 SYS SYSTA00  SBS SBSA00

 SYN0032I  TOTAL CATALOG SEGMENTS DELETED   FOR ENVIRON ENVA    :        2

 SYN0033I  TOTAL CATALOG SEGMENTS PROCESSED FOR ENVIRON ENVA    :        7


 SYN0031I  ENVIRONMENT ENVZ     CATALOG TO MCF ELEMENT SYNCRONIZATION STARTING.


 SYN0032I  TOTAL CATALOG SEGMENTS DELETED   FOR ENVIRON ENVZ    :        0

 SYN0033I  TOTAL CATALOG SEGMENTS PROCESSED FOR ENVIRON ENVZ    :        1


 SYN0034I  TOTAL CATALOG SEGMENTS PROCESSED, ALL ENVIRONMENTS:         113
```

If the utility is executed again in update mode, no discrepancies should be found, as indicated in the following sample report.

```
1  (C) 2005 Computer Associates International, Inc.                                    10NOV04 14:36:29       PAGE   1

                                        Endevor MCF Catalog Synchronization UTILITY LOG             RELEASE 7.0  SERIAL B7000C




 SYN0000I  BEGINNING PHASE 1


 SYN0002I  ENVIRONMENT ENV1     MCF ELEMENT TO CATALOG SYNCRONIZATION STARTING


 SYN0003I  TOTAL CATALOG SEGMENTS CREATED FOR ENVIRON ENV1    :        0
```

```
SYN0004I  TOTAL CATALOG SEGMENTS UPDATED FOR ENVIRON ENV1    :        0

SYN0005I  TOTAL #  OF ELEMENTS PROCESSED FOR ENVIRON ENV1    :      102


SYN0002I  ENVIRONMENT ENVA     MCF ELEMENT TO CATALOG SYNCRONIZATION STARTING



SYN0003I  TOTAL CATALOG SEGMENTS CREATED FOR ENVIRON ENVA    :        0

SYN0004I  TOTAL CATALOG SEGMENTS UPDATED FOR ENVIRON ENVA    :        0

SYN0005I  TOTAL #  OF ELEMENTS PROCESSED FOR ENVIRON ENVA    :        5


SYN0002I  ENVIRONMENT ENVZ     MCF ELEMENT TO CATALOG SYNCRONIZATION STARTING



SYN0003I  TOTAL CATALOG SEGMENTS CREATED FOR ENVIRON ENVZ    :        0

SYN0004I  TOTAL CATALOG SEGMENTS UPDATED FOR ENVIRON ENVZ    :        0

SYN0005I  TOTAL #  OF ELEMENTS PROCESSED FOR ENVIRON ENVZ    :        1


SYN0011I  TOTAL # OF ELEMENTS PROCESSED, ALL ENVIRONMENTS:          108



SYN0000I  BEGINNING PHASE 2



SYN0031I  ENVIRONMENT ENV1     CATALOG TO MCF ELEMENT SYNCRONIZATION STARTING.


SYN0032I  TOTAL CATALOG SEGMENTS DELETED   FOR ENVIRON ENV1    :        0

SYN0033I  TOTAL CATALOG SEGMENTS PROCESSED FOR ENVIRON ENV1    :      102


SYN0031I  ENVIRONMENT ENVA     CATALOG TO MCF ELEMENT SYNCRONIZATION STARTING.
```

```
SYN0032I  TOTAL CATALOG SEGMENTS DELETED   FOR ENVIRON ENVA    :        0

SYN0033I  TOTAL CATALOG SEGMENTS PROCESSED FOR ENVIRON ENVA    :        5


SYN0031I  ENVIRONMENT ENVZ     CATALOG TO MCF ELEMENT SYNCRONIZATION STARTING.


SYN0032I  TOTAL CATALOG SEGMENTS DELETED   FOR ENVIRON ENVZ    :        0

SYN0033I  TOTAL CATALOG SEGMENTS PROCESSED FOR ENVIRON ENVZ    :        1


SYN0034I  TOTAL CATALOG SEGMENTS PROCESSED, ALL ENVIRONMENTS:         108
```

Appendix C: Interfacing with CA Common Services

This section contains the following topics:

Formatting CA Endevor SCM Messages

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

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

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

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

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

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

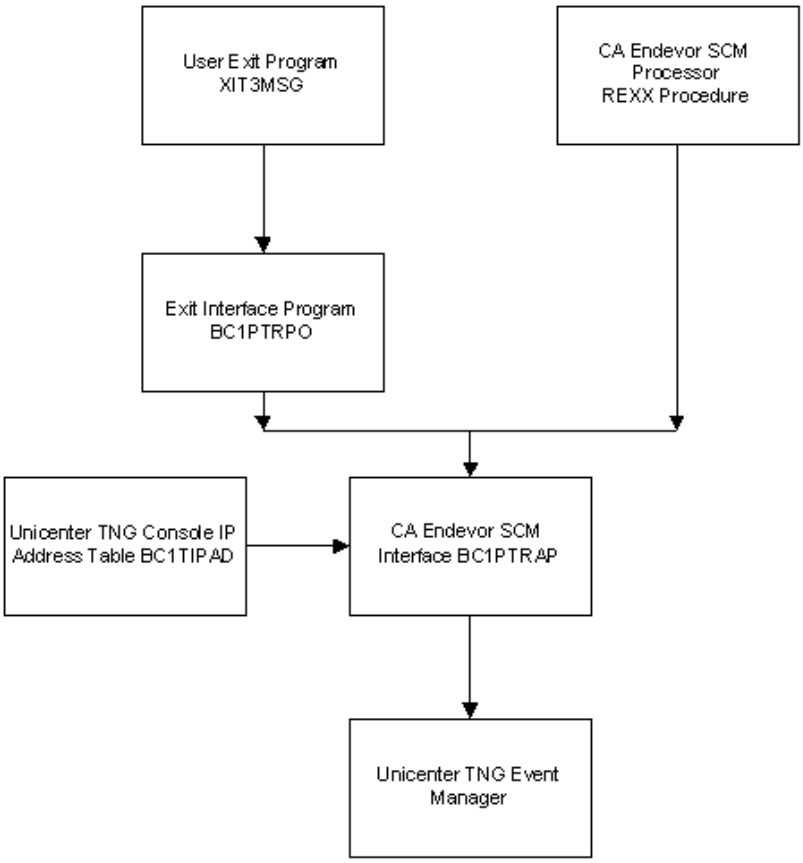
CCS allows a 102-byte message field to be displayed on the Event Console. When you construct a message to be sent to CCS, consider including the following information:

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

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

How the Interface Works

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



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

Note: For more information about completing these tasks, see the CCS tutorials and documentation.

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

Calling the Interface from a User Exit

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

The BC1PTRPO user exit interface program requires the following two parameters:

Message

A102 byte message field which is passed 'as-is' to the Event Console.

Result-area

An 80-byte field into which BC1PTRPO returns the result of the Event submission. If the Event submission is successful, it contains "OK" as the first two characters, padded with spaces. Otherwise, it contains the reason for the Event submission failure.

The user exit must build the message and interrogate the result area. Writing user exits assumes you have a working knowledge of CA Endevor SCM user exit architecture. A sample user exit is delivered as member XIT3MSG, in the iprfx.iqual.CSIQOPTN library.

Note: For more information about user exits, see the *Exits Guide*.

Calling the Interface from a Processor

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

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

Message

A 102 byte message field which is passed 'as-is' to the Event Console.

Result-area

A field where BC1PTRAP returns the result of the Event submission. If the Event submission is successful, it contains "*-ok" as the first four characters, padded with spaces. Otherwise, it contains the reason for the Event submission failure.

The REXX procedure must build the message and interrogate the result area.

Sample Processor Fragment

To call the CA Endevor SCM Interface from a CA Endevor SCM processor, you should refer to the sample processor fragment shown next, in combination with the REXX procedure sample, ENFSAMP, that follows.

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

Sample REXX Procedure

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

The following REXX procedure corresponds to the previous processor fragment:

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

Appendix D: Long Names and USS Supported Files

This section contains the following topics:

Long Name and USS Support

Long name support allows CA Endevor SCM to support the following:

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

The USS Path Name

The USS path name can be a maximum of 768 characters long and contain the following characters:

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

USS Files

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

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

The following are the rules for naming a file:

- The file name cannot contain null or / (slash) character.
- The file name does not support double- byte characters.
- Shells are case-sensitive, so distinguish between upper and lower case characters. For example, FILE1 and file1 are not the same file names.
- The file name can include the following:
  - Suffixes
  - An extension, consisting of a period (.) and several characters, to indicate its file type. Files containing C code could have the extension .c, for example:

    ```
    dbmod3.c
    ```

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

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

Element Names

Element names can be 255 characters long, and may contain the following characters:

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

Long Name Support and CA Endevor SCM Interfaces

Long name support varies for the CA Endevor SCM interfaces. The following information explains the differences:

ISPF

No support for actions involving long name elements or path names.

Quick Edit

Element names greater than 10 characters are truncated in lists. The truncated format consists of:

- A left brace ([)
- The first five characters of the element name
- An ellipsis (…)
- A right brace (])

For example, element 'longnameelement' displays in ISPF as:

```
\{longn...\}
```

Note: Element information is not available from the ISPF list; it is only accessible from CA CMEW or the Eclipse-Based UI. If there are multiple long name elements and the first five characters are identical, ISPF displays a single entry.

HFS path names longer than 44 characters are truncated in type definitions and on the element master display screen. The truncated format consists of:

- Left bracket ({)
- The first 39 characters of the path name
- An ellipsis (...)
- Right bracket (})

Batch (SCL), API, CA CMEW, Eclipse-Based UI, and Web Services User-Written Client Programs

- Element names can be 255 characters long, containing mixed-case
- Periods, underscores, and hyphens are valid characters
- Path names can be 768 characters, not including the HFS file name

HFS path and file name support is unavailable for the following functions:

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

Long Name Elements

This section provides information about how long name elements are added, stored, and displayed in CA Endevor SCM.

Adding and Retrieving Long Name Elements

These file types are used to add elements to CA Endevor SCM or retrieve elements from CA Endevor SCM:

- DSN files-CA Endevor SCM supports base and source output libraries:
  - PDS, PDSE
  - ELIB
  - CA Panvalet
  - CA Librarian
  - HFS
- USS path and file names

  Elements can be retrieved from CA Endevor SCM and saved as a DSN member or as a file in the USS directory. There is no relationship between the element name and the member name/file name. The element name's characteristics limit which file type can be used as the target of the retrieve action, as shown in the following table:

Element Name Characteristics	USS	DSN
Long-alphanumeric mixed case and @, $, #,., - ,_	Yes	No
Short-alphanumeric mixed case and @, $, #,., - ,_	Yes	No
Short-alphanumeric upper case and @, $, #,., - ,_	Yes	Yes

Note: Long element names are greater than 10 and less than 256 characters. Short element names are less than or equal to 10 characters.

Storing Long Name Elements

The following table summarizes the supported storage definitions and any limitations for elements in CA Endevor SCM:

Element Name Characteristics	USS Base	DSN Base	USS Source Output	DSN Source Output
Long-alphanumeric mixed case and @, $, #,., - ,_	Yes	Forward-delta only	Yes	No
Short-alphanumeric mixed case and @, $, #,., - ,_	Yes	Forward-delta only	Yes	No
Short-alphanumeric upper case and @, $, #,., - ,_	Yes	Yes	Yes	Yes

Note: Long element names are greater than 10 and less than 256 characters. Short element names are less than or equal to 10 characters.

Displaying Element Information

Element lists behave differently in the various interfaces. ISPF and Quick Edit lists use brackets to designate long name elements and short name elements with lowercase characters. CA CMEW and the Eclipse-Based UI display the full element name. The following table explains these differences:

Element Name Characteristics	Sample Element Name	ISPF/Quick Edit	CA CMEW or Eclipse-Based UI
Long-alphanumeric mixed case and @, $, #,., - ,_	abcdefghijklm	{abcde…}	Abcdefghijklm
Short-alphanumeric mixed case and @, $, #,., - ,_	LoNgNaMe	{LoNgNaMe}	LoNgNaMe
Short-alphanumeric uppercase and @, $, #,., - ,_	ELEMENT1	ELEMENT1	ELEMENT1

Note: Long element names are greater than 10 and less than 256 characters. Short element names are less than or equal to 10 characters. Element information is *not* available from the ISPF or Quick Edit lists. It is only available from CA CMEW or the Eclipse-Based UI. If there are multiple long name elements and the first five characters are identical, the ISPF list displays a single entry.

USS Files and Directories

This section provides information about USS files and directories.

USS Directories and CA Endevor SCM Libraries

With UNIX System Services (USS) file support, some CA Endevor SCM libraries can be located on this file system. Currently, the only supported libraries are:

- Base
- Source output
- Include

The following table lists the CA Endevor SCM libraries that can be located on USS file systems:

Library	z/OS	USS
Master Control Files	Yes	No
Package data set	Yes	No
Base libraries	Yes	Yes
Delta libraries	Yes	No
Processor output libraries	Yes	No
Source output libraries	Yes	Yes
Processor load libraries	Yes	No
Processor listing libraries	Yes	No
CA Endevor SCM listing libraries	Yes	No
Include libraries	Yes	Yes

Note: Processors can write outputs to USS directories. In addition, ensure that the maximum record size associated with the USS file can be accommodated by the delta library specified on the type definition.

USS Directories and Type Definitions

If an USS file is to be used as a base file no types should share this file, meaning that any type definition that uses an USS file as a base should keep the base file unique. Also, element registration (using Processor Group Output Types) should be used to enforce the segregation across types.

When designating USS directories as base, source output, or include libraries in type definitions, you can use site symbolics, CA Endevor SCM symbolics, or text to construct paths. When viewing the type definition in ISPF, symbolics are displayed for the longer paths. As with CA Endevor SCM or user symbolics, the site symbolics are resolved at runtime.

USS Directories and CA Endevor SCM Actions

CA Endevor SCM supports USS directories for Add, Update, and Retrieve actions initiated in batch (SCL), or from CA CMEW or the Eclipse-Based UI. The following table summarizes the actions supported by USS locations:

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

USS Files and CA Endevor SCM Actions

CA Endevor SCM allows you to work with USS files, long name elements and short names containing lower case characters using CA CMEW, the Eclipse-Based UI, or the batch interface. The following table summarizes the supported actions:

Action	CA CMEW or Eclipse-Based UI.	ISPF Quick Edit	Batch SCL
Master display, Element name	Displays full name	N/A	{abcde…}

Action	CA CMEW or Eclipse-Based UI.	ISPF Quick Edit	Batch SCL
Master display, 'Retrieved to'	Displays full path	N/A	Brackets if > 44 characters
Master display, 'Added from'	Displays full path	N/A	Brackets if > 44 characters
Add/Update, 'Add from'	Input allowed	Input *not* allowed	Full HFS path and file name
Retrieve, 'Retrieve to'	Input allowed	Input *not* allowed	Full HFS path and file name
Type definition: Base libraries, Source output libraries, Input libraries	Input allowed	Input *not* allowed	Using site symbolics
Type display	Displays full path	Displays Symbolics	N/A

The HFS RECFM Field

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

Note: The change of a delimiter for types with existing elements may make these elements unusable for Retrieve actions using CA Endevor SCM, CA CMEW,or the Eclipse-Based UI.

HFS RECFM (COMP/CR/CRLF/CRNL/F/LF/NL/V)

Identifies the record delimiter used in a HFS file. A record delimiter is necessary due to the nature of HFS files. HFS files contain one large data stream. Therefore, a delimiter is used to identify individual records within that data stream.

This is also true for elements associated with a type defined as text (Data Format=T). When the element is transferred from CA Endevor SCM to CA CMEW or any other client application that uses CA Endevor SCM WebServices. These programs recognize and append the delimiter for the exchanged records. However, these delimiters are then translated from EBCDIC to the target code page. When the element is transferred to CA Endevor SCM, all delimiters are removed before translation. The one exception is where two delimiters appear consecutively, in which case a single space is inserted to avoid creating zero length records.

If a delimiter is not specified, the system defaults to NL.

Note: The change of a delimiter for types with existing elements may make these elements unusable for Retrieve actions.

Acceptable delimiter values are the following:

COMP— Variable length records compressed by CA Endevor SCM.

CR— Carriage return. The hex value is x'0D' (and also x'0D' in ASCII).

CRLF— Carriage return \ line feed. The hex value is x'0D25' (or x'0D0A' in ASCII). This is the default delimiter used on the Windows platform.

CRNL— Carriage return \ new line. The hex value is x'0D15' (or x'0D0A' in ASCII).

Note: CRNL is sometimes referred to as *CRLF*. However, CRNL should not be confused with the HFS RECFM value CRLF, which is the EBCDIC version.

F— Fixed Length. When the type is specified as fixed length, CA Endevor SCM does not expect or use any delimiters found in the file. This option should also be used for *Stream data* especially in binary elements, to allow the native file system (ASCII or EBCDIC) delimiters to remain unchanged.

LF— Line feed. The hex value is x'25' (or x'0A' in ASCII).

NL— Default. New line character. The hex value is x'15' (or x'0A' in ASCII). This is the delimiter used by the OEDIT and OBROWSE editors. It is also the default delimiter on Unix platforms and is supported in several Windows editor environments, making it a good choice for interoperability.

V— Variable. The first four bytes of the record contain the RDW (record descriptor word). The RDW contains the length of the entire record, including the RDW.

Note: The maximum record length is 32000 bytes.

File Permissions Given to Output UNIX Files

The file permissions given to output UNIX files are dependent on various factors as described next:

- CA Endevor SCM managed files (base files and source output files)
  - If option ENABLE_ALTID_USS_SECURITY is enabled, then the file permissions specified in that option are used.
  - Otherwise, the UNIX permissions of 777 (a=rwx) are set.

  Note: For more information about ENABLE_ALTID_USS_SECURITY, see the *Security Guide*.
- Other output files
  - Files written to by the RETRIEVE command
    - If the target file already exists, its existing file permissions are kept.
    - If the target file does not exist, the UNIX permissions of 777 a=rwx are set.
    - The directory structure for the target output file must give all users execute access.

- Files written to by CONWRITE during processor execution.
  - If the DD statement for the file contains file permissions ons (PATHMODE) values, those permissions are used.
  - If the DD statement does not contain file permission and if the file is an existing file, the existing permissions are kept.
  - If the DD statement does not contain file permission and If the file does not exist, the UNIX permissions of 777 (a=rwx) are set.
  - The directory structure for the target output file must give all users execute access.
- Files written by the ENUSSUTL utility, which collects package backout information for USS processor output files.

Concurrent Access ENQUEUE Scheme

To manage concurrent use of the CA Endevor SCM footprint file when it is in a z/OS UNIX directory, CA Endevor SCM issues an ENQUEUE for the file.

The ENQ QNAME and RNAME follow the convention used by ISPF Edit and OEDIT as documented in the IBM *ISPF Planning and Customization* manual. The QNAME is SPFEDIT. RNAME is a 12-byte value consisting of the 4-byte Inode#, the 4-byte Device# and a 4-byte sysplex indicator. The sysplex indicator is x'00000001' if z/OS UNIX is in a sysplex (OextSysplexActv is set ON) or x'00000000' if z/OS UNIX is not in a sysplex (OextSysplexActv is set OFF). The ENQ scope is set to SYSTEMS if z/OS UNIX is in a sysplex, otherwise it is set to SYSTEM.

USS Supported Files and the Alternate ID

When using a USS base library, CA Endevor SCM source management accesses the library using the alternate ID. CA Endevor SCM also uses the alternate ID when it accesses a USS source output library in the CA Endevor SCM reserved processors BASICGEN and BASICDEL.

The security context of a processor step is determined by the following factors:

- **ALTID keyword—**The security context used for USS file access in processor steps is determined by the ALTID keyword. Processors can run under either the credentials of the user, or under the Alternate ID, so that USS outputs (created, copied, compiled, and so on) will have the Alternate ID as the owner or group owner. For more information about Alternate ID support, see How to Activate the Alternate ID for Data Set Protection, in the *Security Guide*.
- **PATHOPTS=(OCREAT)—**Any USS data sets created in processor steps using PATHOPTS=(OCREAT) are created using the security context of the user ID, prior to the invocation of the processor program. When used in the processor step, these data sets are opened using the security context of the processor step.
- **EXEC PGM=BPXBATCH—**The security context of a processor step executing the IBM USS utility program BPXBATCH depends on the following factors:
  - Whether BPXBATCH runs a shell script (PARM='SH') or directly executes an executable file (PARM='PGM'). *We recommend BPXBATCH be invoked directly as processor step.*
  - The settings of the Environment variables _BPX_BATCH_SPAWN and _BPX_SHAREAS.

  The following table shows combinations of these parameters and settings with the resulting security context of the processor step executing BPXBATCH.

BPXBATCH parameter	_BPX_BATCH_SPAWN	_BPX_SHAREAS	Security context
SH	ANY	ANY	Alternate ID
PGM	NO	n/a	Alternate ID
PGM	YES	NO	Alternate ID
PGM	YES	YES	User ID

- **BPXBATSL, BPXBATA2 and BPXBATA8—**These programs always run under the context of the user ID, because they process requests in the same manner as the last row in the prior table (_BPX_BATCH_SPAWN=YES, and _BPX_SHAREAS=YES).
- **BPXBATCH in an IKJEFT01 processor step—**

  Note: We do not recommend the use of BPXBATCH in an IKJEFT01 step, because the results can be unpredictable.

  Whether BPXBATCH invoked by a CLIST or REXX in a processor step executing IKJEFT01 executes under the context of the alternate ID depends on the following two *additional* factors:

  - Whether or not a LGNT$$$I swap was performed prior to the invocation of BPXBATCH.
  - Whether or not a prior USS service call was made in the job step prior to the BPXPBATCH call. For example, the prior USS service call could occur when a USS Base or Source Output Library was used during the CA Endevor SCM job step, or when a shell script was executed in this or any prior CA Endevor SCM action.

  The following table shows the security context results from these two additional factors when calling BPXBATCH from a processor step that executes IKJEFT01:

LGNT$$$I swap performed	Prior USS Service call made	Security context
Yes	No	Alternate ID
Yes	Yes	Failure
No	No	User ID
No	Yes	User ID

Appendix E: Using Email Notification

This section contains the following topics:

Email Notification

Email notification within CA Endevor SCM is enabled by creating the mainframe ID and email ID table, ESMTPTBL. This table maps mainframe user IDs to email IDs or group names. When an email notification event occurs, the invoked program scans the ESMTPTBL table for an appropriate match. The program searches the table for a matching mainframe ID. If a match is found, an email addressed to the associated email ID is sent. If no match is found, no email is sent.

However, if you code the DFTID=USERID parameter in the ESMTPTBL, instead of bypassing unlisted IDs, CA Endevor SCM will construct an email address with the CA Endevor SCM user ID suffixed with the default domain. If your site has defined email aliases for all your mainframe user IDs, coding of DFTID=USERID allows you to reduce the size of the ESMTPTBL and makes it easier to maintain.

The email interface utility BC1PMLIF allows you to write simple email programs without having an in-depth knowledge of the NOTIFY block or the ESMTP table lookup.

Note: For more information about BC1PMLIF, see the *Utilities Guide*. For a sample COBOL email program that uses the BC1PMLIF interface, see the *Exits Guide*.

Email Notification Components

The following components are provided as part of the Email Notification Facility:

$ESMTP

Macro that builds a table (ESMTPTBL) that maps mainframe IDs to email IDs.

ESMTPTBL

Table created by the macro $ESMTP to map mainframe IDs to email IDs.

BC1JTABL

Sample JCL used to assemble and link any of the Tables, including the ESMTPTBL table.

The $ESMTP Macro

The macro, $ESMTP, builds the mainframe ID and email ID table, ESMTPTBL. The macro has a section for global information and a section for each mainframe user ID.

The JCL to assemble and link this macro is located in iprfx.iqual.CSIQJCL. The resulting load module, ESMTPTBL, must reside in the CA Endevor SCM authorized library (uprfx.uqual or iprfx.iqual.CSIQAUTH).

The global information section includes the following fields:

ATSIGN

Use this field to specify the byte value that replaces the @ sign in email addresses. You can specify any value, except 00 and 40. For example, ATSIGN=80. The default value is 7C.

HOSTNAME

NJE node name of the z/OS system where SMTP is running.

DFTDOMAIN

Domain name to be used in email addresses. You can override this field in the mainframe ID section.

MAILFROM

Defines a default MAIL FROM value. The value is free form and CA Endevor SCM neither does syntax checking on the parameter nor appends a hostname to it.

The MAIL FROM value can be defined in several ways. The value that is used is the one that is found first in the following search order:

1. NOTFOPT@ (if NOTFOPT has been set to 2 or 3)
2. NOTFROM@
3. MAILFROM
4. The default CA Endevor SCM value

DFTURL

URL where CA CMEW is running. You can override this field in the mainframe ID section.

DFTID=USERID

If CA Endevor SCM cannot find the USERID in ESMTPTBL and DFTID=USERID is coded, CA Endevor SCM attempts to send an email to the mainframe user ID by constructing an email address with the CA Endevor SCM user ID suffixed with the default domain. Without DTFID=USERID, CA Endevor SCM does not send an email if it cannot find the mainframe user ID in ESMTPTBL. This parameter does not override email addresses in ESMTPTBL. If an invalid address is found in ESMTPTBL, no email is sent to that user ID.

To enable the default ID feature, only one value is permitted (DFTID=USERID). If the value is omitted, or coded as null (DFTID=), then the parameter has no effect.

SMTPTASKNAME

The name of the SMTP address space. The default is SMTP and is rarely subject to change.

SMTPCLASS

The SYSOUT CLASS associated with the SMTP task. The default is B and is rarely subject to change.

The mainframe ID/email ID section includes the following fields:

HOSTNAME

NJE node name of the z/OS system where SMTP is running.

MFID

The mainframe user ID or package approver group name. This field can contain up to 16 character.

EMAILID

The portion of the email ID that precedes the @. This field size is unlimited.

DOMAIN

The portion of the email ID that follows the @. If not specified, the global default domain is used. This parameter supports an environment where email IDs might be defined in multiple domains.

URL

The URL where CA CMEW is running. If not specified, the global default URL is used. This parameter enables you to point users to different implementations of CA CMEW.

Disable Notification for Specific Users

To completely disable notifications for a particular user, simply provide a dummy or invalid email address. The default does not override any explicitly coded email address; therefore, if you leave an invalid email address in ESMTPTBL, no email is actually delivered to that particular user.

Sample ESMTPTBL

A sample ESMTPTBL table is provided in the Tables library. The sample ESMTPTBL includes the DFTID=USERID parameter, which enables CA Endevor SCM to construct email addresses from the CA Endevor SCM user ID and the default domain. If your ESMTPTBL includes DFTID=USERID, when a package is cast requiring an approval from one or more users not defined in the table, for example peter01 and sally02, CA Endevor SCM will send an email to peter01@yourco.com and sally02@yourco.com, in addition to emails to users defined in the table.

Sample BC1JTABL

The JCL that can be used to assemble and link-edit tables, including ESMTPTBL, is located in member BC1JTABL, provided in the JCL library (*iprfx.iqual.CSIQJCL*). An alternative is to employ an SMP/E USERMOD to accomplish this.

Synchronization Notification

The *Synchronization Notification* feature immediately sends an email to the element change owner, when an element is changed or added higher in the software lifecycle map and is found to be out-of-sync with the element located lower in the map.

Out-of-sync elements are instances of the same element with the element at the higher location in the software lifecycle map having change levels that are not reflected in the element located lower in the map. These elements are out of synchronization (out of sync).

A *source sync check* occurs whenever an element is changed or arrives at a new location in the lifecycle map. All instances of that element located lower in the software lifecycle map are checked to determine if the newly changed or added element is out of synchronization (out of sync) with the elements located lower in the map. The source sync check verifies the last level time stamp of the element at the higher lifecycle location against the base level time stamp of the lower elements. If the dates do not match, a source management's synchronization check function is invoked for these elements to determine if they are actually out of sync.

Provided Synchronization Notification has been enabled by the administrator, it is initiated by the following activities:

- The element actions: Add, Update, Move, Restore, Transfer C1-C1, and Transfer from Archive.
- The Quick Edit actions: Create and Edit.
- The Search and Replace utility.

For each out-of-sync element found, an email is sent to the owner associated with the last source change level of the element located lower in the map than the newly changed or added element. However, if a change level does not exist yet, the email is sent to the last element action user ID. In addition, a message is written to the action log to record the out-of-sync elements. In foreground, if an action causes out-of-sync elements, the "route sync problem" text appears on the action's panel, unless this message is suppressed by the administrator.

Note: If email notification is **not** already set up at your site, the administrator must set it up in order for this feature to work. For details see Email Notification. (see page 309)

Enable Synchronization Notification

To let users know when an element is added up the software lifecycle map that is out of synchronization with their element located lower in the map, the administrator can enable synchronization notification.

Note: If email notification is **not** already set up at your site, the administrator must set it up in order for this feature to work. For details, see Email Notification. (see page 309)

To Enable Synchronization Notification

1. In the Type=Main section of the C1DEFLTS table, set the SYNC@CHK parameter to Y as follows:

   SYNC@CHK=Y,

   Source synchronization is enabled.

   Note: To turn off source synchronization, set one of the following. (SYNC@CHK=, is the default.)

   SYNC@CHK=N,

   SYNC@CHK=,

2. Set the severity level of the out-of-sync messages written to the action log, by setting SYNC@MSV= parameter in the C1DEFLTS table to one of the options.

   SYNC@MSV=[*I,W,C*],

   I- Informational message.

   W-Warning message.

   C-Caution message.

   Blank-Default. If no option is specified (for example, SYNC@MSV=,) then the option defaults to I for informational message.

   When an out-of-sync element is added, a message with the severity level you specify is written to the action log.

3. Mark an environment as the starting location in the software lifecycle where you want source sync checks to be performed. In the Type=Environment section of the C1DEFLTS table, set the SYNC@BLOC parameter to Y (SYNC@BLOC=Y,) for this environment. The options for this parameter are as follows:

   SYNC@BLOC=[Y,N],

   Y- The environment is a source synchronization location.

   N- Default. The environment is **not** a source synchronization location.

   Blank- If no option is specified (for example, SYNC@BLOC=,) then the option defaults to N.

   Specifies whether the Environment, Stage location is a source synchronization location. If SYNC@BLOC=Y, when an action adds or moves an element to this location, or to any location higher up the map, a sync check is performed. If an element located lower in the map is not in sync with the new element, a synchronization notification email is sent to the owner of the out-of-sync element at the lower location. If more than one SYNC@BLOC is specified in C1DEFLTS, regardless of its specification (SYNC@BLOC=, SYNC@BLOC=N, or SYNC@BLOC=Y,) all SYNC@BLOC parameters after the first one in the map are ignored.

Example: Synchronization Notification - Starting Point in the Software lifecycle Map

Assume that your site's C1DEFLTS table maps environment DEV to QA1, maps QA1 to QA2, and maps QA2 to PRD. If the C1DEFLTS table definition of QA2 has SYNC@BLOC set to Y, then QA2 and PRD are synchronization locations. It is not necessary to mark PRD.

Source Synch Check Email Format

A separate email notification is given for each out-of-sync element. The email is formated as follows. The "Package : package" text line is omitted if the element is not associated with a package.

```
User id has just placed element name type name version level vvll in SCM at location:
```

```
   Env          : environment
   System     : system
   Subsystem: subsystem
   Stage ID   : id
   With CCID: ccid         Comment: comment
   Package   : package

Your element vvll at location below is now out-of-sync with the above element.
   Env          : environment
   System     : system
   Subsystem: subsystem
   Stage ID   : id
   With CCID: ccid         Comment: comment
   Package   : package
```

View Synchronization Notification Settings

There are several ways to check the C1DEFLTS settings in effect at your site for the Synchronization Notification function.

- To see if source sync checking is enabled and what severity level is assigned to the action log messages, you can check the site options using either of the following methods:
  - See the panel Site Information from C1DEFLTS. The Function Control section of the panel includes the following fields: Sync Sev Msg and Sourc Sync Check.
  - Run a Site Options Settings report, which lists the Source Sync Chk and Sync Sev Msg fields in the C1DEFLTS section of the report.
- To see if an environment is an active source synchronization location, you can use either of the following methods:
  - See the Environment Information panel's field Out-of-Sync Alerts.
  - Use the CSV utility's List Environment function and view the SYNC LOC field in the returned data.
- If your site uses the API, the API List Site and List Environment requests return Source Notification settings.

Appendix F: Using the Trace Facilities

This section contains the following topics:

How to Enable Trace Facilities

The trace facilities help you isolate problems by monitoring selected processes. When a trace facility is active, information is recorded in a system trace output file.

Important: The trace formats of all traces are subject to change at any time. Therefore, we recommend that you *do not* use trace output in a production setting or write programs that interpret the trace output. Also, you should be selective in which traces you enable, because activating traces that are not needed results in unnecessary overhead.

To enable a trace facility, use the following syntax depending on whether you want to trace batch or foreground processing:

- For foreground, use:

  `alloc f(DDname) sysout(*)`

- For batch, use:

  `//BSTERR DD SYSOUT=*`

  `//DDname DD SYSOUT=*`

To route the output of the trace to a data set, replace sysout as follows:

- For foreground, use:

  `alloc f(DDname) da(dsname)`

- For batch, use:

  `//BSTERR DD DSN=dsname,DISP=SHR`

  `//ddname DD DSN=dsname,DISP=SHR`

  Set the DSN attributes for the trace DSN as follows:

  `LRECL=133,BLKSIZE=1330,RECFM=FB,DSORG=PS`

DDnames

Specifies the name of the trace. The following traces are valid:

EN$TRALC

Traces dynamic allocation.

EN$TRAUI

Traces the Alternate ID. Also traces logon and logoff information.

EN$TRESI

Traces ESI (Endevor External Security).

EN$TRFPV

Traces component validation.

EN$TRITE

Traces if-then-else evaluation.

EN$TRLOG

Traces logon and logoff information.

EN$TROPT

Traces Endevor table definitions.

EN$TRSMF

Writes SMF records. This trace requires the following dsname:

```
//EN$TRSMF DD DISP=SHR,DSN=....,LRECL=27994,BLKSIZE=27998,RECFM=VB
//*                          Writes SMF records to //EN$TRSMF.
//*                          Requires DCB or existing file !
```

EN$TRSYM

Traces symbolic resolution.

EN$TRXIT

Traces Exits.

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W