CA Endevor® Software Change Manager

Utilities Guide

Version 17.0.00

Fourth Edition

CA Technologies Product References

This document references the following CA Technologies products:

- CA ACF2 ™
- CA Common Services™ (CA Common Services)
- CA Endevor® Software Change Manager (CA Endevor SCM)
- CA Endevor® Software Change Manager Automated Configuration (CA Endevor SCM Automated Configuration)
- CA Librarian® Base for z/OS (CA Librarian Base)
- CA Panvalet® for z/OS (CA Panvalet)
- CA Top Secret® (CA Top Secret)

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

- OPTIONS Parameters Common to All CSV Functions (see page 243)— Added an important note about using the NOCSV option.

Version 17.0, Third Edition

- Using the BSTXCOPY Utility (see page 333)— Added to describe this utility, which is similar to the CA Endevor SCM processor utility BSTCOPY, except that BSTXCOPY executes outside of a processor.

Version 17.0, Second Edition

- Alter Action Metadata (see page 270)— Added to describe the Alter action metadata fields extracted by the List Element option of the Comma Separated utility.

Version 16.0, Third Edition

For Version 16.0 PTF RO66163, January 2014:

- Reload Control Card (see page 228)— Updated to add the option Retain Processor History.

Version 16.0, Second Edition

- The BC1PNCPY Utility (see page 38)— Updated to add a caution note about the truncation of member names longer than eight characters.

Version 16.0

- Reorganize Element Catalog and Eindex Files (see page 24)— Added to describe the BC1JRCAT utility.
- The Browse Panel (see page 57)— Updated to change the release and level format to VVLL.
- The Email Interface Program BC1PMLIF (see page 145)— Updated to clarify the description for the MYSMTP-URL parameter, which Includes the default URL value of ESMTPTBL in the email notification.

- OPTIONS Parameters Common to All CSV Functions (see page 243)— Updated the description for the NOCSV option to clarify that the NOCSV output does not include all the fields that appear in the output when this option is *not* specified.
- Extracted Destination Data (see page 256)— Updated to add fields to the CSV output for USS host and remote path names: HOST USS PREFIX, DISP, REMOTE USS PREFIX, and DISP. Also, updated the data set field names DISP, UNIT, and VOLSER to HOST DSN DISP and REMOTE DSN DISP, HOST DSN UNIT and REMOTE DSN UNIT, and HOST DSN VOLSER and REMOTE DSN VOLSER.
- Extracted SMF Activity Data (see page 303)— Updated to add the field AUTOGEN SPAN. Also updated to add the following environment information fields for target locations: ENV NAME (T), STG NAME (T), STG ID (T), STG # (T), SYS NAME (T), SBS NAME (T), TYPE NAME (T), ELM NAME (T), ELM VV (T), ELM LL (T).
- Extracted Type Data (see page 321)— Updated to add the field Element RECFM.
- CSV Utility JCL (see page 324)— Updated to clarify that the CSV utility must shut down the API at the end of the calling program.

Release 15.1

- Allocate and Initialize a VSAM ELIB Data Set (see page 43)—Updated to add a note that the ELIB data set format supports z/OS VSAM extended addressability for VSAM data sets.

Version 15.0

- Processing Methods for Replacing Phrases (see page 82)—Added to describe Expand Includes utility processing methods for the REPLACING phrases on LIBRARIAN -INC or COBOL COPY statements.
- Select a Processing Method for Replacing Phrases (see page 82)—Added to describe how to select a processing methods for REPLACING phrases.
- How the Expand Includes Utility Specifies INCLUDE Libraries (see page 85)—Updated to specify that the ddnames limit is raised from 100 (00-99) to 204 (00-99, AA-DZ)
- How the INCLUDE Library is Searched (see page 86)—Updated to indicate that numeric names will be searched ahead of AA-DZ names.
- Extracted Destination Data (see page 256)—Added IPNAME and IPPORT fields.
- Component Base Information (see page 266)—Added CMPNT BASE VV.
- Extracted Package Data
  - Package Backout/Backin Information (see page 283)—Updated to add the element action backout value (E) to the Pkg Backed Out flag.
- Extracted Subsystem Data (see page 314)—Updated to add EXCLUDE FROM PROC O/P CK.
- Extracted System Data (see page 317)—Updated to add PROC O/P REG ACROSS SBS.

- Extracted Type Data (see page 321)—Updated to change the output heading for the delta format to FWD/REV/IMG/LOG ELM DELTA.
- Unused Processor Symbolics Override Utility—Modified last bullet item to reflect changes to BC1PSDEL utility.
- List and Remove Unused Processor Symbolic Overrides (see page 330)—Updates made to reflect changes to BC1PSDEL utility.

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.

Contents

Chapter 10: Using the Point in Time Recovery Feature 147

Chapter 11: Using the Search and Replace Utility 171

Chapter 1: File Definition and Maintenance

This section contains the following topics:

CA Endevor SCM Data Sets

The following describes the CA Endevor SCM data sets in detail:

Master Control File

Definitions of stages, systems, subsystems, element types, and elements. This file is accessed and updated by CA Endevor SCM to perform source and output management, and to handle other miscellaneous services. There is one Master Control File (MCF) for each stage at a site. The Master Control Files are defined as a function of installation.

Package data set

Package information for all environments defined for the site.

Base and delta libraries

Source statements for elements, including processors. The base library contains the source as originally added to CA Endevor SCM. The delta library contains the changes made to the elements. These libraries are specified separately to each element type definition, but can be shared by multiple element types. The libraries can be shared across systems and stages, but make sure that the defined record length for each library is adequate to store the element source from all systems/stages that share the library. At a minimum, you must have one library to store base and delta members in each environment.

If you use CA Endevor SCM ACM, the component base and delta are also stored in these same libraries and require a logical record length of at least 259. Each base or delta library can be an OS partitioned data set (PDS or PDS/E), a CA Panvalet data set, a CA Librarian data set, or a self-reorganizing ELIB data set.

Source output library

Latest full source form of each element, created during output management. This is an optional library defined to each element type (although the same library can be shared across element types). This library is designed for use with COBOL copybooks, assembler macros, or JCL procedures that are copied elsewhere (and therefore have to be available in full source form). It can be used for any type element however.

The source output library can be an OS PDS, or a CA Panvalet or CA Librarian data set.

CA Endevor SCM listing library

Listings output by the CA Endevor SCM CONLIST utility or by the type PROCESS generate processor (GPPROCSS). For both listing purposes, this library must be specific to a particular stage, but can be shared across systems. This library can be an OS PDS or PDS/E or ELIB data set.

Processor load library

Load module form of each processor defined to CA Endevor SCM, as output by the type PROCESS generate processor, GPPROCSS. This library is specific to a particular stage, but can be shared across systems. This library must be an OS load library (RECFM=U).

Note: In addition to the libraries referenced, you may also have INCLUDE libraries and user libraries (copy libraries, macro libraries, JCL libraries, and so forth) defined to processors. These libraries are not specific to CA Endevor SCM, and therefore are not described here.

How to Define the CA Endevor SCM Files

You establish your CA Endevor SCM files during installation. The following instructions explain how to set up additional base and delta libraries, processor listing libraries, processor load libraries, and source output libraries, as well as how to monitor and maintain the existing libraries.

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

Note: For more information about establishing CA Endevor SCM files during installation, see the *Installation Guide*.

ELIB Data Sets

ELIB data sets offer several performance advantages over OS PDS. In particular, ELIB data sets:

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

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

Note: For more information about ELIB data sets, see the chapter "Setting Up ELIB Data Sets (see page 29)."

How to Define Base and Delta Libraries

Base and delta libraries can be ELIB (ELIB) data sets, partitioned data sets (PDS or PDS/E), or CA Panvalet or CA Librarian data sets. Depending upon the delta format chosen, each base/delta library set can be any combination of these file types.

For example, if you use reverse deltas, a regular PDS or PDS/E to be used to store element base, and an ELIB dataset to store deltas. Base libraries can be HFS directories.

Base and Delta Library Space Requirements

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

Note: For more information about determining the disk space required by a base or delta library, see the *Installation Guide*.

Allocate a New PDS or PDS/E

To allocate a new PDS or PDS/E, use ISPF/PDF option **3** (Utilities), option **2** (Data sets), or any suitable IBM utility (such as IEFBR14).

Specify the following DCB, assigning a block size appropriate to your disk device:

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

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

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

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

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

Allocate a New CA Panvalet or CA Librarian Data Set

To allocate a new CA Panvalet or CA Librarian data set, see the appropriate CA documentation.

Allocate and Initialize a New ELIB Data Set

To allocate and initialize a new ELIB data set, see the chapter "Setting Up ELIB Data Sets (see page 29)."

Define a CA Endevor SCM Listing Library

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

Note: For more information about determining the disk space required for a CA Endevor SCM listing data set, see the *Installation Guide.*

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

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

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

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

Note: For more information about allocating a new ELIB data set, see the chapter "Setting Up ELIB Data Sets (see page 29)."

Define a Processor Load Library

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

Note: For more information about determining the disk space required for a CA Endevor SCM listing data set, see the *Installation Guide.*

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

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

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

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

Define a Source Output Library or HFS Directory

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

Note: For more information about determining the disk space required for a CA Endevor SCM listing data set, see the *Installation Guide*.

To define a source output library or HFS directory

Use ISPF/PDF option **3** (Utilities), option **2** (Data sets), or any suitable IBM utility (such as IEFBR14).

Specify the following DCB, assigning a block size appropriate to your disk device:

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

or

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

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

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

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

How to Maintain Files

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

Monitor Space Utilization

The Master Control File and Package data sets are VSAM files, and should be maintained using the standard IBM VSAM maintenance utility IDCAMS. To obtain file utilization statistics, run the utility with the LISTCAT command.

Note: For more information about IDCAMS, see your IBM documentation.

You should monitor your CA Endevor SCM libraries regularly, as explained in the following:

OS PDS or PDS/E libraries

Use ISPF/PDF, option **3** (Utilities), option **2** (Data sets) to display space utilization statistics. OS PDS or PDS/E libraries include the CA Endevor SCM listing libraries and processor load library, and may include the base and delta libraries and the source output library.

Note: For more information, see your ISPF documentation.

CA Librarian or CA Panvalet libraries

The base library, delta library, and source output library may be CA Librarian or CA Panvalet files.

Note: For more information, see the appropriate CA Librarian or CA Panvalet documentation.

ELIB data sets

Run the BC1PNLST utility program. ELIB data sets can be used for the base, delta, and/or listing libraries.

Note: For more information about the BC1PNLST utility, see the chapter "Setting Up ELIB Data Sets (see page 29)."

How to Expand or Compress a File

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

Expand or Compress the Master Control File

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

Expand or Compress Package Data Sets

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

Reorganize Element Catalog and Eindex Files

For Element Catalog and Eindex files, run the job supplied as member BC1JRCAT in your installation JCL library. Use this job to reorganize the Element Catalog and Eindex files.

Expand or Compress PDS or PDS/E Libraries

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

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

Expand or Compress CA Librarian and CA Panvalet Libraries

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

Expand or Compress ELIB Data Sets

ELIB data sets are automatically maintained each time a member is stored or updated and therefore, compression is unnecessary. In addition, ELIB data sets can automatically expand into secondary extents.

Note: For more information about changing the secondary expansion quantity or the directory size, see the chapter "Setting Up ELIB Data Sets (see page 29)."

How to Back Up CA Endevor SCM

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

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

The following describes how to back up various CA Endevor SCM files:

The Master Control File

Use the IDCAMS utility with the REPRO command.

A PDS or PDS/E

Use a standard IBM utility (for example, IEBCOPY), or any other appropriate utility in use at your site.

A CA Panvalet or CA Librarian file

See the appropriate documentation.

Back Up VSAM ELIB Data Sets

To back up ELIB VSAM data sets, use the IDCAMS utility and supply the appropriate space and DCB information, and is located in the member BC1JELIB is supplied in your iprfx.iqual JCL library for this purpose. You can also use any other appropriate utility in use at your site.

Back Up BDAM ELIB Data Sets

To back up an ELIB BDAM data set, use the standard IBM IEBGENER utility or any other appropriate utility in use at your site.

How to Back Up Using the Unload, Reload, and Validate Utility

If you do not want to use the standard IBM backup utility, you can use the CA Endevor SCM Unload, Reload, and Validate utility. This utility provides a backup, reload, and validation mechanism for CA Endevor SCM VSAM files (Master Control File and package data set) and base/delta libraries.

Note: For more information about using this utility, see the chapter "Using the Unload, Reload, and Validate Utility (see page 217)."

How CA Endevor SCM Recovery Works

If a critical file is lost or its integrity compromised you must restore all the critical files, including the Master Control File, base library, and delta library. First restore the files using the latest backup. Then reapply any updates made since the last backup.

Important! Keep the source version of all changes made to critical files until a backup can be made. Otherwise, you won't be able to reapply these updates.

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

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

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

How to Recover Using the Unload, Reload, and Validate Utility

You can use the CA Endevor SCM Unload, Reload, and Validate utility for recovery. This utility provides a backup, reload, and validation mechanism for CA Endevor SCM VSAM files (Master Control File and package data set) and base/delta libraries.

Note: For more information about this utility, see the chapter "Using the Unload, Reload, and Validate Utility." (see page 217)

Name Masking

To help you more easily find information and process requests, you can use *name masking*. By substituting a name with the asterisk wildcard character (*), a character with the percent sign placeholder (%), or by using both together, it is much faster and easier to find information and process requests.

Note: For more information about name masking, see the *User Guide*.

SCL Statement Syntax Conventions

When you write SCL statements and commands for elements, environment definitions, and packages, CA Endevor SCM uses the IBM standard to represent the syntax.

Note: For information about syntax, how you code syntax,and sample syntax diagrams, see the *SCL Reference Guide*.

Chapter 2: Setting Up ELIB Data Sets

This section contains the following topics:

ELIB Data Sets

An Endevor LIB (ELIB) is a high performance alternative to OS partitioned data sets under CA Endevor SCM. You can organize CA Endevor SCM base, delta, and listing libraries as ELIB data sets. Three concepts are important when working with ELIBs. These concepts are page size, record format, and target directory page.

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

  ELIB data sets allow you to allocate additional directory pages spontaneously, using the BC1PNLIB utility.

CA Endevor SCM provides three utilities for use when setting up and maintaining ELIB data sets. The following lists these utilities and the purpose of each:

BC1PNLIB

Set up, maintain, and print information about ELIB data sets.

BC1PNLST

Print information about ELIB data sets.

BC1PNCPY

Copy library members from and to ELIB data sets, and to copy members from and to any CA Endevor SCM library format.

Note: The ELIB utilities BC1PNLIB, BC1PNCPY and BC1PNLST cannot be executed within a CA Endevor SCM processor.

The BC1PNLIB Utility

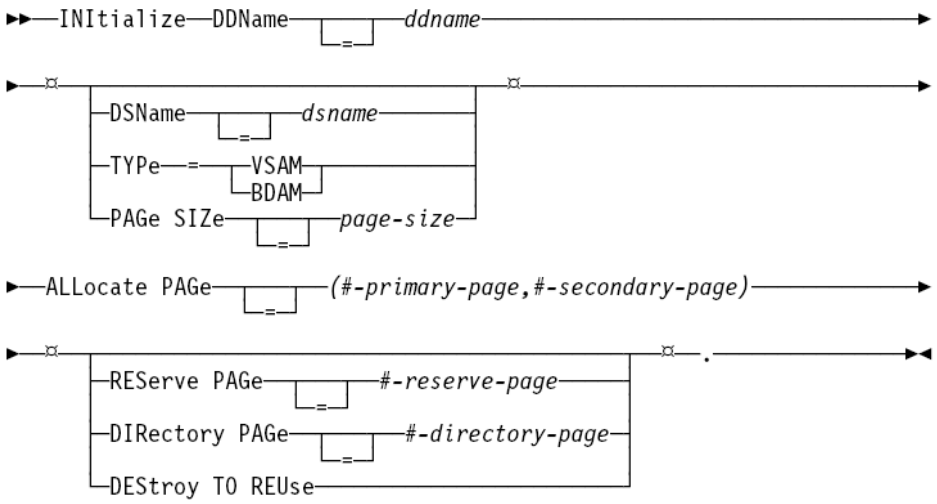
The BC1PNLIB utility allows you to set up and maintain the ELIB data sets. It will also allow you to perform five actions against ELIB data sets:

A Sample JCL to execute the BC1PNLIB ELIB utility is provided with the CA Endevor SCM installation files as member BC1JNLIB in the installation JCL library.

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

The Initialize Function

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



The following is a list of keywords used in the INITIALIZE syntax:

INITIALIZE

(Required) Indicates that you want the BC1PNLIB utility to initialize a ELIB data set. You must have already allocated space for this data set.

DDNAME

(Required) The DDname of the data set you want to initialize.

DSNAME

(Optional) Data set name for this ELIB data set. If you code the DSNAME statement, the system validates the DSname in this statement against the DSname in the JCL.

TYPE

(Optional) Indicates whether this ELIB is a VSAM cluster or a BDAM data set. If you do not code a TYPE statement, the BC1PNLIB utility reads this information from the JCL.

PAGE SIZE

(Optional) If you are using VSAM clusters, page size is system-defined by the IDCAMS utility as eight bytes less than the control interval size.

If you are using BDAM data sets, page size will equal block size. Block size is specified in the JCL. If you also code a PAGE SIZE statement, the system will validate it against the block size you have specified in the JCL.

If you code a PAGE SIZE statement, the entry can have up to six numeric characters.

ALLOCATE PAGES

(Required) Determines number of pages in this ELIB data set. You can specify two numbers:

- #-primary-pages--Indicates the number of pages initialized as primary storage for this ELIB data set.
- #-secondary-pages--Indicates number of pages initialized as secondary storage during each automatic expansion of the ELIB data set.

If you do not want to assign a secondary allocation quantity, do not specify secondary pages.

Both the primary and the secondary entries can be up to six numeric characters.

Note: For more information about estimating space requirements, see see the chapter "Setting Up ELIB Data Sets (see page 29)."

RESERVE PAGES

(Optional) Determines when the system allocates a secondary storage block to this ELIB data set. Expressed as a number of pages (up to six numeric characters). When this number of pages remain unused within the current storage allocation, CA Endevor SCM automatically attempts to allocate a secondary storage block.

- If you assign a RESERVE PAGES value, you should assign a value that is greater than the number of pages you expect the largest data set member to take up. Very often, assigning a value equal to one cylinder of storage is adequate.
- If you omit this clause, the reserve threshold defaults to 1/16 of the number of pages allocated to primary storage. For example, if the primary allocation is 400, the reserve threshold will default to 25 (400/16).
- If you assign a value of 0, ELIB will not automatically expand. In this situation, once the primary allocation is filled any attempts to store a new member will fail until you manually expand the library.

Note: If ELIB tries to expand the library and runs out of space, it automatically sets the reserve threshold value to zero. For more information about resolving this problem, see Expand or Compress ELIB Data Sets (see page 25).

DIRECTORY PAGES

(Optional) Number of target directory pages. The default allocation is 7 pages.

If you want to allocate a different number of target directory pages, enter that number (must be greater than 7) in this statement. Can be up to six numeric characters.

When estimating the number of target directory pages bear in mind that:

- Each directory member requires 84 bytes.
- Each directory page has 32 bytes reserved.

This means that if your page size is 4096 bytes, a target directory page could hold [(4096-32)/84] = 48 members.

Note: If a member is added and the directory page is full, the page overflows, eventually degrading performance. To avoid overflow, allocate target directory pages so that member directory information fills less than 50 percent of the available target directory page space.

Example: Your target directory page size is 4096 bytes, and you want to store directory information for approximately 24 members per page (half of the maximum of 48 members). If you estimate that 1,000 members will reside in the data set, you should allocate the data set with 1,000/24 = 42 target directory pages.

If you omit this clause, the BC1PNLIB utility automatically calculates the number of pages within the ELIB data set needed for its directory. It does this by allocating enough target directory pages to support one member per each 4 pages of the primary allocation.

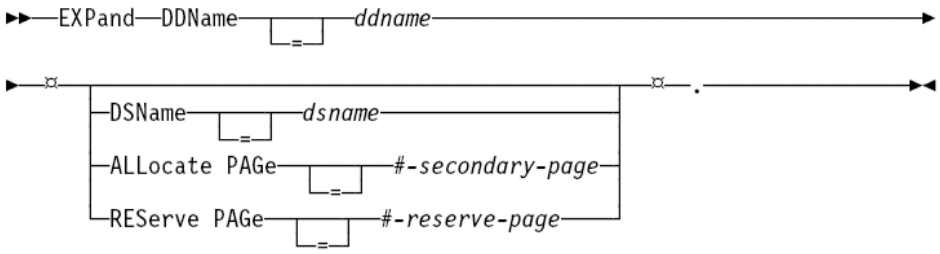
DESTROY TO REUSE

(Optional) Code this statement only if you are re-initializing an ELIB data set.

Note: If you code this statement, all data in the existing data set will be lost.

The Expand Function

Normally, ELIB expands automatically according to your secondary allocation value. The EXPAND function of the BC1PNLIB utility allows you to expand ELIB data sets manually. The syntax for the EXPAND function of the BC1PNLIB utility is shown next:



The following describes the keywords used in the syntax for the EXPAND function:

EXPAND

(Required) Indicates that you want the BC1PNLIB utility to acquire secondary storage for an ELIB data set.

DDNAME

(Required) The DDname of the data set you want to expand.

DSNAME

(Optional) Data set name for this ELIB data set. If you code the DSname statement, the system validates the DSname in this statement against the DSname in the JCL.

ALLOCATE PAGES

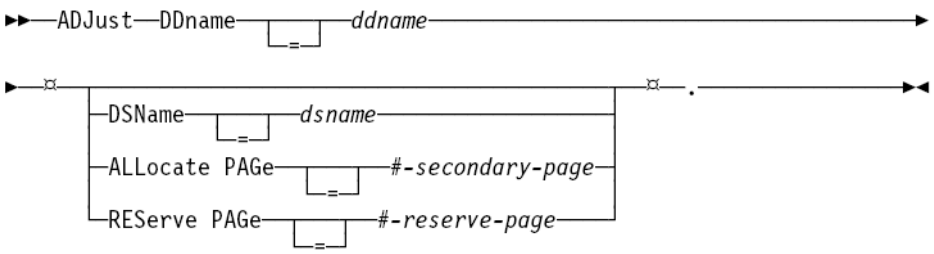
(Optional) Determines number of additional pages in the expanded ELIB data set. If you do not code this statement the secondary allocation originally specified during the INITIALIZE function will be used.

RESERVE PAGES

(Optional) Allows you to respecify the RESERVE threshold established in the INITIALIZE function for this library. If you want to keep the existing RESERVE threshold, do not code this statement.

The Adjust Function

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



The following describes the keywords used in the syntax for the ADJUST function:

ADJUST

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

DDNAME

(Required) The DDname of the data set you want to adjust.

DSNAME

(Optional) Data set name for this ELIB data set. If you code the DSname statement, the system validates the DSname in this statement against the DSname in the JCL.

ALLOCATE PAGES

(Optional) Determines number of pages to be allocated in subsequent secondary allocations.

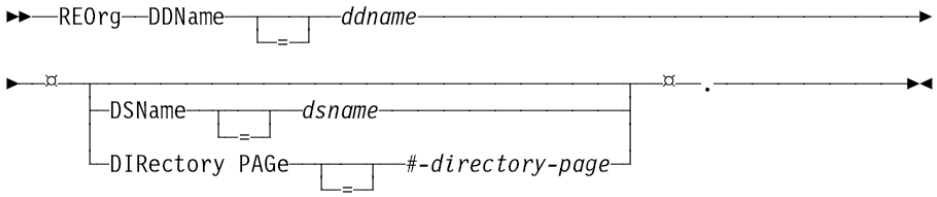
RESERVE PAGES

(Optional) Allows you to respecify the RESERVE threshold established in the INITIALIZE function for this library. If you want to keep the existing RESERVE threshold, do not code this statement.

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

The Reorganize Function

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



The following describes the keywords used in the syntax for the REORG function:

REORG

(Required) Indicates that you want the REORG function of the BC1PNLIB utility to respecify the number of pages allocated to the directory of an ELIB data set.

DDNAME

(Required) The DDname of the data set you want to reorganize.

DSNAME

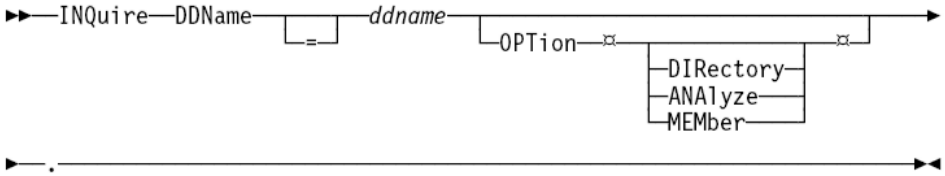
(Optional) Data set name for this ELIB data set. If you code the DSname statement, the system validates the DSname in this statement against the DSname in the JCL.

DIRECTORY PAGES

(Required) Indicates the number of pages you want to allocate for the directory of the ELIB data set specified by the DDname.

The Inquire Function

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



The following describes the keywords used in the syntax for the INQUIRE function:

INQUIRE

(Required) Indicates that you want to use the INQUIRE function of the BC1PNLIB utility.

DDNAME

(Required) The DDname of the data set(s) for which you want to print information.

OPTION

(Optional) Allows you to specify additional reporting options. You can use any or all of these options whenever you run the INQUIRE function of the BC1PNLIB utility. If you omit this clause, CA Endevor SCM prints only data set header information. There are three options:

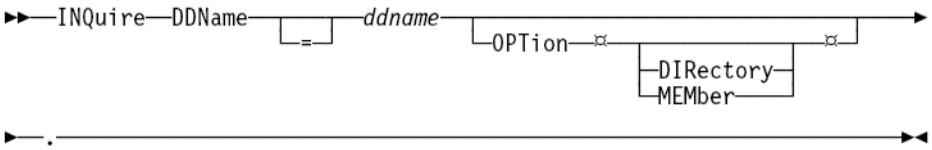
- DIRECTORY--Tells CA Endevor SCM to print information about target directory page utilization.
- MEMBERS--Tells CA Endevor SCM to print footprint information plus member size in pages, records, and bytes.
- ANALYZE--Tells CA Endevor SCM to print an analysis of data set integrity.

The BC1PNLST Utility

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

A Sample JCL to execute the BC1PNLST ELIB utility is provided with the CA Endevor SCM installation files as member BC1JNLST in the installation JCL library.

The following is the BC1PNLST syntax:



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

INQUIRE

(Required) Indicates that you want to use the INQUIRE function of the BC1PNLST utility to retrieve information about an ELIB data set.

DDNAME

(Required) The DDname of the data set(s) you want to inquire against.

OPTION

(Optional) Allows you to specify additional reporting options. You can use any or all of these options whenever you run the INQUIRE function of the BC1PNLST utility. If you do not want additional detail, you can omit this clause entirely; only summary library information will be printed. There are two options:

- DIRECTORY--The DIRECTORY option displays the number of members for each directory page, as well as summary directory usage statistics.
- MEMBERS--The MEMBERS option lists each member of the library, with full CA Endevor SCM footprint information plus member size in pages, records, and bytes.

The BC1PNCPY Utility

The BC1PNCPY utility allows you to copy between any CA Endevor SCM supported library types, while preserving the original CA Endevor SCM footprint of copied members. You can use this utility to copy all or selected members of an existing base, delta, or listing library from a PDS or PDS/E to a ELIB or vice versa. A Sample JCL to execute the BC1PNCPY ELIB utility is provided with the CA Endevor SCM installation files as member BC1JNCPY in the installation JCL library.

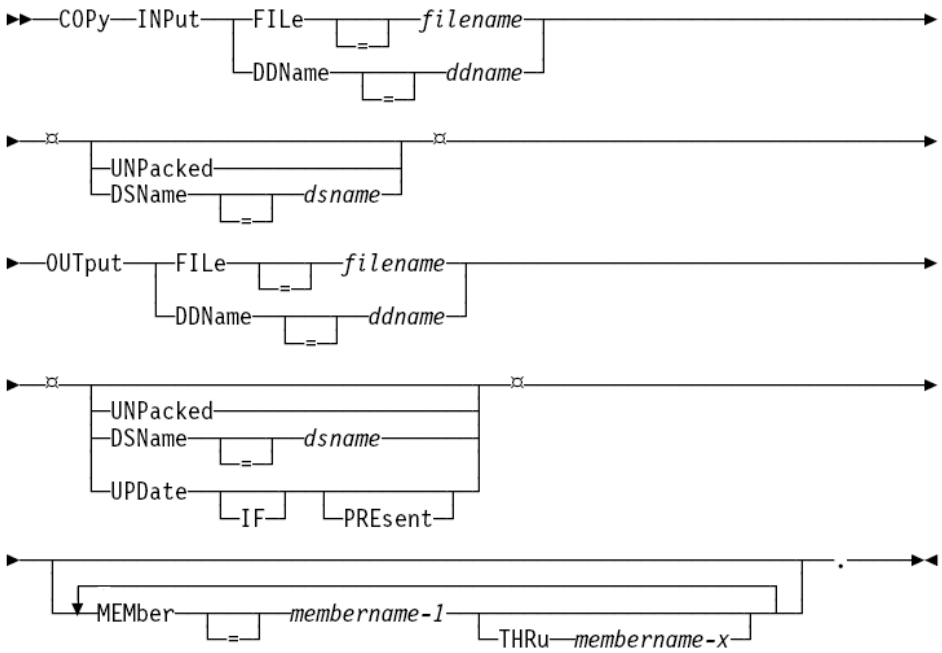
When dealing with large libraries, BC1PNCPY takes a great deal of time to run. In addition, it requires exclusive control of the input and output libraries, for integrity.

Caution: When an element name is longer than eight characters and you copy to a PDS or PDS/E, the element name is truncated at eight characters. When element name truncation occurs, different input members can truncate to the same input name.

If the UPDATE IF PRESENT option is specified, the last duplicate member will overwrite all prior duplicates.

If the UPDATE IF PRESENT option is **not** specified, only the first encountered duplicate member will be overwritten. Any subsequent duplicate member names will not overwrite the first duplicate member.

The following is the BC1PNCPY syntax:



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

COPY

(Required) Indicates that you want to use the BC1PNCPY utility.

INPUT FILE \ DDNAME

(Required) Identifies the source file or DDname.

UNPACKED

(Optional) The UNPACKED option is used with PDS or PDS/Es only, and only if a PDS or PDS/E is not a CA Endevor SCM compressed library. When you use this option with the INPUT statement BC1PNCPY will not attempt to decompress the members being read from the input data set.

DSNAME

(Optional) If you use the DSNAME option, the data set name of the input library will be validated against the DDNAME in the JCL. If the DSN contains periods, it must be enclosed in single quotes.

OUTPUT FILE \ DDNAME

(Required) Identifies the target file or DDname.

UNPACKED

(Optional) The UNPACKED option is used with PDS or PDS/Es only, and only if a PDS or PDS/E is not a CA Endevor SCM compressed library. When you use this option with the OUTPUT statement BC1PNCPY will not attempt to compress the members being written to the output data set.

DSNAME

(Optional) If you use the DSNAME option, the data set name of the target library will be validated against the DDNAME in the JCL. If the DSN contains periods, it must be enclosed in single quotes.

UPDATE IF PRESENT

(Optional) Use this option to update (replace) members with the same name within the target library. If you do not use this option, the utility will not replace members of the same name.

MEMBER ...[THRU ...]

The MEMBER clause(s) limits the COPY request to specific members only. You can specify either a single member or a range of members (using the THRU clause). Note that when entering MEMBER and THRU clauses, you can use the standard CA Endevor SCM wildcard capability.

Note: A period must be coded at the end of each complete statement. If you specify any MEMBER [THRU] clauses, you must enter the period after the last of those clauses. If you specify more than one MEMBER [THRU] clause, do **not** code a comma between the clauses.

By default BC1PNCPY's parser stack supports 1000 MEMBER= statements. If more than 1000 statements are needed, the parameter MEMBERS can be passed to the utility using the PARM= field on the EXEC statement:

```
//JS10        EXEC PGM=NDVRC1,PARM='BC1PNCPYMEMBERS=nnnnnn'
```

Where nnnnnn is a 1 to 6 digit number specifying the approximate number of MEMBER= statements coded. When present, the parser stack is increased to 'nnnnnn' entries. If the specified value is 1000 or less, the default value of 1000 is used.

Note: Specifying this parameter may increase the storage requirements for the step. Approximately 40 bytes per member is needed. Be sure to specify an appropriate REGION size to accomodate the requests.

How to Allocate and Initialize an ELIB Data Set

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

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

Select an Access Method for ELIB Data Sets

You must decide what physical format you want to use for the ELIB data sets; you can use either VSAM or BDAM. Your choice of access method depends on your site's preferences. The following describes the features of VSAM and BDAM access methods:

- VSAM provides better performance by exploiting 31-bit storage and better data set protection mechanisms. VSAM also supports spanned volumes
- BDAM provides slightly better performance and simpler installation and de-installation, but spanned volumes are not supported.

Estimate Space Requirements for ELIB Data Sets

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

- For an ELIB base library, figure out how many members are in the library, and how much space they take up in the data set after the data set has been compressed. Then add 20 percent to this figure, for expansion.
- For an ELIB delta library, estimate approximately half the size of the corresponding base library. If you are an existing CA Endevor SCM user, you may want to use a larger percentage, to accommodate for the growth of your delta libraries over time.
- For an ELIB listing library, estimate 80% of the space required for the corresponding members in a PDS or PDS/E (measured after the PDS has been compressed).

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

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

How to Allocate and Initialize the Data Set

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

Allocate and Initialize a BDAM ELIB Data Set

To allocate a BDAM ELIB data set, use a JCL stream similar to the following:

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

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

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

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

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

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

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

```
//ELIBINIT EXEC PGM=BC1PNLIB
//STEPLIB  DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTU
//     DD DISP=SHR,DSN=iprfx.iqual.CSIQLOAD

//*
//* OMIT THE FOLLOWING DCB INFORMATION IF VSAM
//*
//BDAMINIT  DD DSN=BDAM.DATASET,
//   DCB=(DSORG=DA,BLKSIZE=4096,LRECL=4096,RECFM=FBS),
//   DISP=(OLD,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN   DD *
INIT   DDNAME = BDAMINIT
     PAGE SIZE = 4096
     ALLOCATE = (750,300)
     RESERVE PAGES = 150
     DIRECTORY PAGES = 14
.
INQUIRE  DDNAME = BDAMINIT
 .
/*
```

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

Allocate and Initialize a VSAM ELIB Data Set

Note: The ELIB data set format supports z/OS VSAM extended addressability for VSAM data sets. The extended addressability feature lets you allocate VSAM data sets that are larger than 4 gigabytes. If extended addressability is enabled in z/OS for any of your VSAM ELIB data sets, CA Endevor SCM recognizes that this option is enabled and handles ELIB access appropriately. You do not need to make any changes in CA Endevor SCM to enable support for extended addressability. Ask your Storage Administrator to determine whether z/OS VSAM extended addressability is available for your site and how you can implement it.

To allocate a VSAM cluster, use the IBM IDCAMS utility in a JCL stream similar to the following:

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

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

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

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

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

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

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

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

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

```
//ELIBINIT EXEC PGM=BC1PNLIB
//STEPLIB  DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTU
//     DD DISP=SHR,DSN=iprfx.iqual.CSIQLOAD

//*
//*
//*
//INITVSAM  DD
//   DSN=VSAM.DATASET,DISP=(OLD,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN   DD *
 INIT   DDNAME = INITVSAM
      PAGE SIZE = 4088
      ALLOCATE = (749,300)
      RESERVE PAGES = 150
      DIRECTORY PAGES = 14
.
INQUIRE  DDNAME = INITVSAM
 .
/*
```

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

Reinitialize an ELIB Data Set

To reinitialize an ELIB data set, you can do the following:

1. Delete and reallocate the library
2. Rerun the initialization procedure, with the following additional clause: DESTROY TO REUSE.

   Note: If you use this clause, any data currently in the library will be destroyed. Use this option with caution.

How to Expand ELIB Data Sets

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

The following example shows a sample JCL and the control statements that you can use to accomplish this:

```
//EXPAND  EXEC PGM=BC1PNLIB
//STEPLIB  DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTU
//     DD DISP=SHR,DSN=iprfx.iqual.CSIQLOAD

//INITVSAM  DD DSN=VSAM.DATASET,
//   DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN   DD *
      EXPAND  DDNAME = INITVSAM
      ALLOCATE PAGES = 300
 .
 INQUIRE  DDNAME = INITVSAM
 .
/*
```

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

How to Adjust ELIB Data Sets

You can change the primary, secondary, and reserve threshold space allocations for an ELIB data set by using the ADJUST function of the BC1PNLIB utility.

You can use the ADJUST function to do the following:

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

The following example shows a sample JCL and the control statements that you can use to adjust an ELIB data set:

```
//ADJUST  EXEC PGM=BC1PNLIB
//STEPLIB  DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTU
//     DD DISP=SHR,DSN=iprfx.iqual.CSIQLOAD

//INITVSAM  DD DSN=VSAM.DATASET,
//   DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN   DD *
 ADJUST  DDNAME=INITVSAM
      ALLOCATE PAGES = 600
      RESERVE PAGES = 300
 .
 INQUIRE  DDNAME=INITVSAM
 .
/*
```

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

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

How to Reorganize ELIB Directory Pages

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

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

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

To reorganize a directory, use JCL and control statements similar to the following:

```
//REORG   EXEC PGM=BC1PNLIB

//STEPLIB  DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTU
//     DD DISP=SHR,DSN=iprfx.iqual.CSIQLOAD

//INITVSAM DD DSN=VSAM.DATASET,
//   DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN   DD *
 REORG   DDNAME=INITVSAM
      DIRECTORY PAGES = 20
 .
 INQUIRE  DDNAME=INITVSAM
 .
/*
```

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

How to Print ELIB Data Set Information

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

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

How to Print Data Set Header Information

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

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

```
//INQUIRE  EXEC PGM=BC1PNLST
//STEPLIB  DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTU
      DD DISP=SHR,DSN=iprfx.iqual.CSIQLOAD

//ELIB  DD DSN=ELIB.DATASET,
//   DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN   DD *
      INQUIRE  DDNAME=ELIB
      .
/*
```

How to Print Member Information

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

To print member information, use JCL similar to the following:

```
//INQUIRE  EXEC PGM=BC1PNLST
//STEPLIB  DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTU
      DD DISP=SHR,DSN=iprfx.iqual.CSIQLOAD

//ELIB  DD DSN=ELIB.DATASET,

//   DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN   DD *
      INQUIRE  DDNAME=ELIB
      OPTION
       MEMBERS
      .
/*
```

How to Print Target Directory Page Information

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

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

```
//INQUIRE  EXEC PGM=BC1PNLST

//STEPLIB  DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTU
      DD DISP=SHR,DSN=iprfx.iqual.CSIQLOAD

//ELIB  DD DSN=ELIB.DATASET,
//   DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN   DD *
 INQUIRE  DDNAME=ELIB
      OPTION
       DIRECTORY
      .
/*
```

How to Print Data Set Analysis Information

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

To verify the integrity of a ELIB library, use JCL and control statements similar to the following:

```
//ANALYZE  EXEC PGM=BC1PNLIB
//STEPLIB  DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTU
//     DD DISP=SHR,DSN=iprfx.iqual.CSIQLOAD

//ELIB  DD DSN=ELIB.DATASET,
//   DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN   DD *
 INQUIRE  DDNAME=ELIB
      OPTION
      ANALYZE
      .
/*
```

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

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

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

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

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

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

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

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

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

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

How to Convert To or From ELIB Format

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

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

To copy between CA Endevor SCM libraries or from/to an external library, use JCL and control statements similar to the following:

```
//CONVERT  EXEC PGM=NDVRC1,PARM='BC1PNCPY'

//STEPLIB  DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTU
//     DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTH
//CONLIB  DD DISP=SHR,DSN=iprfx.iqual.CSIQLOAD

//OLDBASE  DD DSN=BST.OLD.BASE,
//   DISP=(OLD,KEEP)
//NEWELIB  DD DSN=BST.NEW.BASE,
//   DISP=(OLD,KEEP)
//SYSPRINT DD SYSOUT=*
//BSTERR  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN   DD *
 COPY   INPUT  DDNAME = OLDBASE
      OUTPUT  DDNAME = NEWELIB
      .
/*
```

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

Chapter 3: Controlling Load Modules

This section contains the following topics:

Module Control

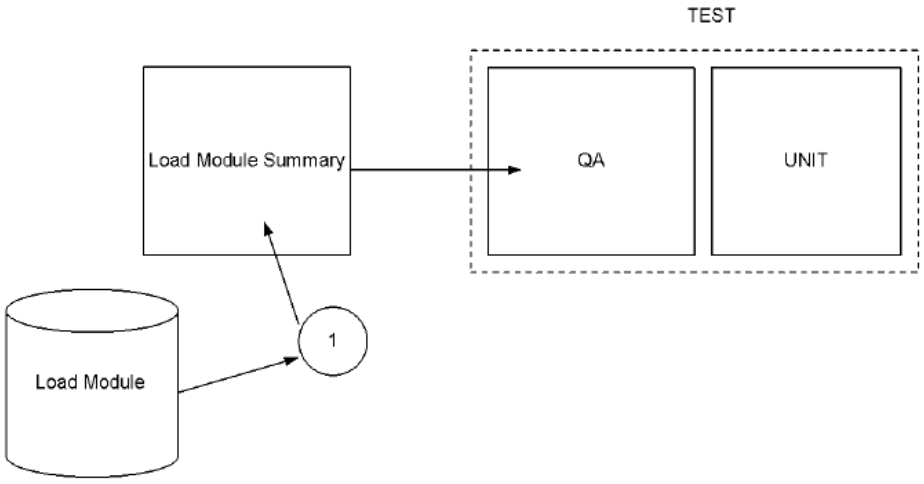
You can place the following modules under the control of CA Endevor SCM:

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

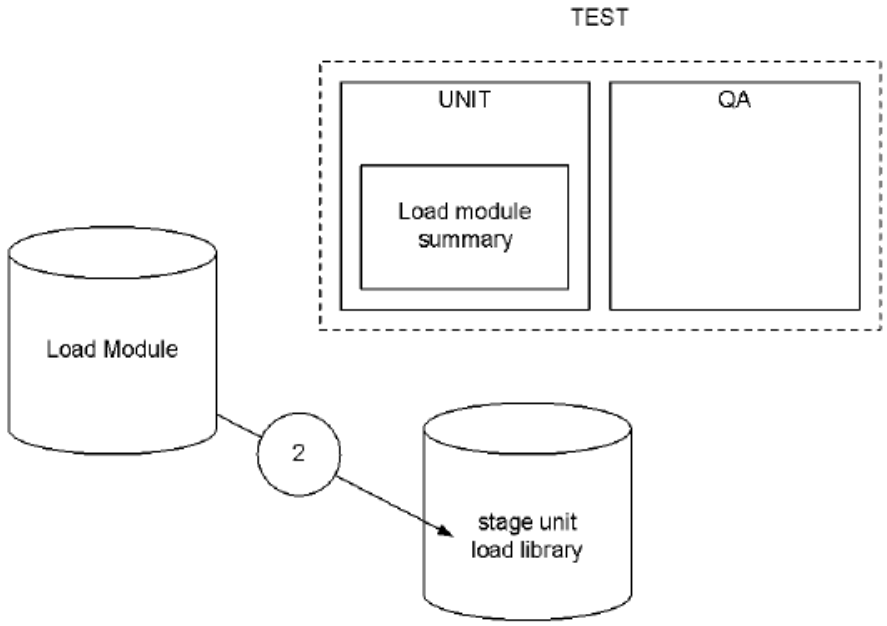
This ability to track load modules allows you to put, for example, vendor code, which is often distributed in load module format only, under CA Endevor SCM's control.

How Controlling Load Modules Works

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



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



Note: For more information about generate processors for load modules, see How to Track Changes to Load Modules (see page 59).

View Load Module Information

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

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

The Browse Panel

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

Link date

Date when the load module was last link edited.

Linkage editor

Identifier for the linkage editor used to link edit the load module.

Entry point offset

Position of entry point.

Size

Size of the load module, in decimal bytes.

EP

Identifies the entry point CSECT.

CSECT

CSECT name and date when it was last generated.

Size

Size of the CSECT, in decimal bytes.

Translator

Identifier for the compiler that performed the translation.

CSECT

CSECT name.

Environ

Footprint environment.

System

Footprint system.

Subsystem

Footprint subsystem.

Element

Footprint element name.

Type

Footprint type.

S

Footprint stage.

VVLL

Footprint version and level for the element.

Date

Footprint date.

No ZAP Data Present

Indicates that no ZAPs have been applied to the current load module.

Note: For more information about how ZAP information is displayed, see The Changes Panel (see page 58).

Attributes

Load module attributes assigned by the linkage editor.

Note: For a description of individual attributes, see your linkage editor documentation.

The Changes Panel

This change panel tells you that:

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

The following describes the ZAP information fields:

ZAP to CSECT

Name of CSECT to which a ZAP was applied.

Date

Date the ZAP was applied to the CSECT.

ID

User-defined ZAP ID.

How to Track Changes to Load Modules

To track changes to load modules using CA Endevor SCM, you must perform the following actions:

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

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

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

Sample Generate Processor

A sample generate processors for ADD, UPDATE, and TRANSFER actions related to load modules is shown next:

```
//ADDCOPY EXEC PGM=BSTCOPY
//    EXECIF=(&C1ACTION.,EQ,ADD)
//SYSPRINT DD SYSOUT=*
//ILIB   DD DSN=&C1USRDSN.,DISP=SHR
//OLIB   DD DSN=user.stg1.loadlib,DISP=SHR,FOOTPRNT=CREATE
//SYSIN  DD *
   C  I=ILIB,0=OLIB
   S  M=((&C1USRMBR.,&C1ELEMENT.,R))
//UPDCOPY EXEC PGM=BSTCOPY
//    EXECIF=(&C1ACTION.,EQ,UPDATE)
//SYSPRINT DD SYSOUT=*
//ILIB   DD DSN=&C1USRDSN.,DISP=SHR
//OLIB   DD DSN=user.stg1.loadlib,DISP=SHR,FOOTPRNT=CREATE
//SYSIN  DD *
   C  I=ILIB,0=OLIB
   S  M=((&C1USRMBR.,&C1ELEMENT.,R))
//XFRCOPY EXEC PGM=BSTCOPY
//    EXECIF=(&C1ACTION.,EQ,TRANSFER)
//SYSPRINT DD SYSOUT=*
//ILIB   DD DSN=ndvr.input.loadlib,DISP=SHR,
//OLIB   DD DSN=ndvr.output.loadlib,DISP=SHR,FOOTPRNT=CREATE

//SYSIN  DD *
   C  I=ILIB,0=OLIB
   S  M=((&C1USRMBR.,&C1ELEMENT.,R))
```

Where:

user.stg1.loadlib

User-defined Stage 1 load library

ndvr.input.loadlib

Load library associated with the from location of the transfer.

ndvr.output.loadlib

Load library associated with the to location of the transfer.

Sample Move Processor

A sample move processor for load modules is shown:

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

Sample Delete Processor

You can use standard delete processors for load module management.

Note: For a sample delete processor, see the *Extended Processors Guide*.

Chapter 4: Using the BSTPCOMP Utility

This section contains the following topics:

The BSTPCOMP Utility

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

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

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

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

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

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

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

How to Control Compare Output

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

The No Overrides Method

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

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

Example: Sample JCL

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

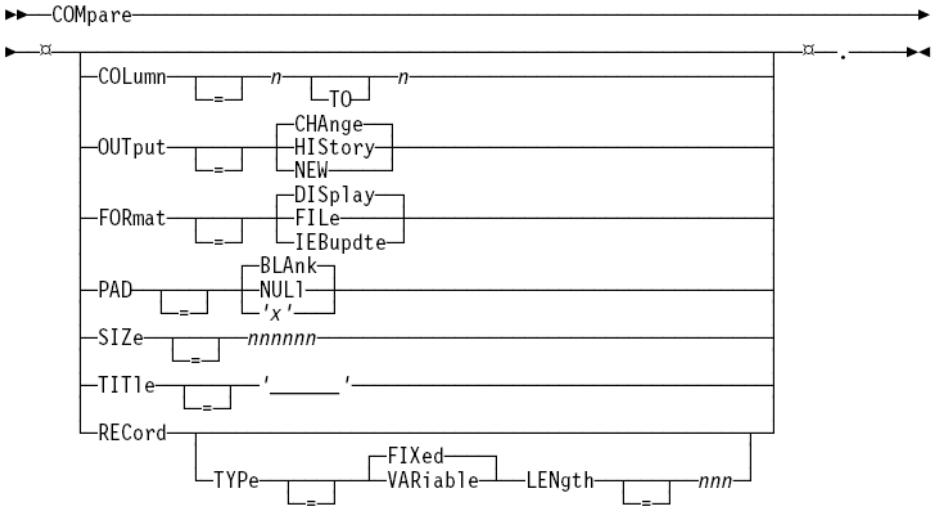
Note: NDVRIN1 and NDVRIN2 must be sequential datasets or a partitioned dataset with a member name specified.

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

The Control Card Execution Method

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

The syntax for the control card is shown next:



Note: For more information about the IEBUPDTE option, see The IEBUPDTE Request Card Generator (see page 71).

A description of the syntax clauses follows:

COLUMN

Specifies the range of columns to compare. If omitted, columns 1-72 are used.

OUTPUT

Specifies which records are to be output. The default is CHANGES only. The other options are: HISTORY (shows the existing member together with both inserts and deletes), and NEW (shows the new member, highlighting inserts only).

FORMAT

Specifies the output format. The default is DISPLAY which writes the output to file NDVRLST. The syntax listing is produced first, followed by the original output file in report format--with carriage control and page headings. The FILE option (that is, BROWSE) writes the data to DDname NDVRPCH without page headings.

PAD

PAD is applicable to variable length records only. The default is BLANK (which pads short records with blanks up to the compare length). The options are NULL (which pads with binary zeros) or “x” (which pads with the specified single character).

SIZE

Specifies a file-size estimate for the sort. By default, this option is ignored.

TITLE

Appears before the first data line as a line of asterisks, followed by the title string and another line of asterisks. Useful if FORMAT=FILE.

RECORD

This clause is provided for DOS-compatibility only.

Example: Control Card JCL

The sample JCL which follows would produce a report:

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

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

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

```
//COMPARE  EXEC  PGM=BSTPCOMP,REGION=2000K
//STEPLIB  DD   DSN=iprfx.iqual.CSIQLOAD,DISP-SHR
//*
//NDVRIN1  DD   DSN=OLD.FILE.LIBRARY(MEMBER),DISP=SHR
//NDVRIN2  DD   DSN=NEW.FILE.LIBRARY(MEMBER),DISP=SHR
//*
//NDVRLST  DD   SYSOUT=*  **SYNTAX LISTING + COMPARE OUTPUT**
//*SORTWORK FILES USED ONLY IF FILES TOO LARGE FOR IN-MEMORY COMPARE
//SORTWK01  DD   UNIT=DASD,SPACE=(TRK,(1,30)
//SORTWK02  DD   UNIT=DASD,SPACE=(TRK,(1,30)
//*
//NDVRIPT  DD   *
 COMPARE COLUMN=1 TO 80 OUTPUT=HISTORY.
//*OUTPUT OF COMPARE: NOTE: RECORD LENGTH=#COMPARE COLUMNS + 16
//NDVRPCH  DD   DSN=OUTPUT.FILE.NAME,DISP=(,CATLG),
// UNIT=DASD,SPACE=(TRK,(5,5),RLSE),
// DCB=(LRECL=88,BLKSIZE=6072,RECFM=FB)
//*
```

The PARM-Controlled Execution Method

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

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

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

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

output-format

The 2-character code that indicates the type of comparison information you want reported (character 1) and the format of the output file (character 2). The default is CD.

Specify the first character (type of information you want) as follows:

- **C-**Print only the **changes** between the two files: that is, those lines that are in file 2 but not in file 1, or those lines that are deleted from file 2.
- **B-**Print **(browse)** the contents of file 2, highlighting those lines that are not in file 1 by printing “%INSERT” to the left of those lines: “%” allows you to scan for changes easily; “INSERT” indicates that the line was new in file 2.

- **H**-Print a **history** of both files, including:
  - The entire contents of file 2, highlighting those lines that are not in file 1 by printing “%INSERT” to the far left.
  - Lines that were in file 1 but not in file 2, highlighting these lines with “%DELETE” to the far left.

Specify the second character (output format) as follows:

- F-The output file is in browse format and does not have any ASA characters or headers. The output is written to DDname NDVRPCH
- **D-**The output file is formatted for print, and includes ASA characters and headers. The output is written to DDname NDVRLST.

from

Starting character for the compare. BSTPCOMP begins its search at this position in both files. The default is 1.

thru

Ending character for the compare. BSTPCOMP ends its search with this position, within both files. For variable-length records, if the record in one file is longer than that in the other, and the thru character extends beyond the end of the record, BSTPCOMP pads according to the pad-char specification before performing the compare.

The default thru specification is 72.

rec-count

Largest number of records in either file. The default is 10000. Estimate high when specifying this value.

pad-char

Pad character used for variable-length records, as described previously for the thru parameter. Specify this as follows. The default is BLANK.

- BLANK--Blanks
- NULL--Null values (binary zeroes)

nnn

The hexadecimal equivalent of nnn, where nnn is a 1-3 character decimal value. Specify 64 to pad with X'40', 255 to pad with X'FF', and so forth.

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

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

Note the coding convention for each variable:

RECFM must be fixed block.

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

Sample Changes Report Output

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

```
COPYRIGHT (C) CA, INC.,               E N D E V O R    mm/dd/yy 11:58:23   PAGE  1
                                          RELEASE X.XX SERIAL XXXXXX
BSTPCOMP - FILE COMPARE UTILITY
 COMPARE OUTPUT=CHANGES .                      00350002
 INSERT MAIN    $FUNCSTG PLSIZE=4              000006
 DELETE MAIN    $FUNCSTG                       000006
 INSERT WORD3   DS        F                    000010
 INSERT         SPACE 3                        000012
 INSERT MAIN999 $FEND                          000014
 DELETE         $FEND                          000012
%****** RECORDS: FILE 1 = 00014 FILE 2 = 00016 INSERTS = 00004 DELETES = 00002 ******
```

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

```
COPYRIGHT (C) CA, INC.,                    E N D E V O R     mm/dd/yy 11:58:27   PAGE  1
                                           RELEASE X.XX SERIAL XXXXXX
BSTPCOMP - FILE COMPARE UTILITY
 COMPARE OUTPUT = NEW  .                     00490002
        COMPTEST TITLE 'PROGRAM TO DEMONSTRATE COMPARE UTILITY'          000001
        COMPT    $MODNTRY LINKAGE=SUB                                    000002
                 TITLE 'DSECTS: DCB '                                    000003
                 DCBD DSORG=PS                                           000004
                 TITLE MAIN: ENTRY FROM CALLER'                          000005
%INSERT MAIN     $FUNCSTG PLSIZE=4                                       000006
        GLOBALS  DS 0D                                                   000007
        WORD1    DS  F                                                   000008
        WORD2    DS  F                                                   000009
%INSERT WORD3    DS  F                                                   000010
                 $FUNC                                                   000011
%INSERT          SPACE 3                                                 000012
                 SR  R15,R15         RETURN CODE = SUCCESS               000013
%INSERT MAIN999  $FEND                                                   000014
                 $MODEND                                                 000015
                 END                                                     000016
%****** RECORDS: FILE 1 = 00014 FILE 2 = 00016 INSERTS = 00004 DELETES = 00002 ******
```

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

```
COPYRIGHT (C) CA, INC.,                      E N D E V O R    mm/dd/yy 11:58:23   PAGE  1
                                           RELEASE X.XX SERIAL XXXXXX
BSTPCOMP - FILE COMPARE UTILITY
 COMPARE OUTPUT = HISTORY .                    00630002
        COMPTEST TITLE 'PROGRAM TO DEMONSTRATE COMPARE UTILITY'          000001
        COMPT    $MODNTRY LINKAGE=SUB                                    000002
                 TITLE 'DSECTS: DCB '                                    000003
                 DCBD DSORG=PS                                           000004
                 TITLE MAIN: ENTRY FROM CALLER'                          000005
%INSERT MAIN     $FUNCSTG PLSIZE=4                                       000006
%DELETE MAIN     $FUNCSTG                                                000006
        GLOBALS  DS 0D                                                   000007
        WORD1    DS  F                                                   000008
        WORD2    DS  F                                                   000009
%INSERT WORD3    DS  F                                                   000010
                 $FUNC                                                   000011
%INSERT          SPACE 3                                                 000012
                 SR  R15,R15         RETURN CODE = SUCCESS               000013
%INSERT MAIN999  $FEND                                                   000014
%DELETE          $FEND                                                   000012
                 $MODEND                                                 000015
                 END                                                     000016
%****** RECORDS: FILE 1 = 00014 FILE 2 = 00016 INSERTS = 00004 DELETES = 00002 ******
```

BSTPCOMP Return Codes

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

3000

The input files are identical for the columns compared. No reports were produced.

3001

An input or output file could not be opened. Ensure that the DD statements are correct for all files and try again.

3002

The number of records in one or both of the input files exceeds the maximum count specified by the rec-count parameter. Increase the count and try again. It is better to estimate high rather than low.

3003

The LRECL for an input file exceeded 256 (260 for variable-length). You cannot use this file as input.

3005

The record format for an input file is Undefined. The record must specify either Fixed or Variable.

3006

An input parameter is missing or invalid (for example, thru >from). Check your syntax against the previous parameter descriptions, correct the problem, and resubmit the job.

3007

BSTPCOMP completed its compare successfully and found differences between the files. This is the standard return code.

The IEBUPDTE Request Card Generator

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

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

How to Generate Control Cards from a CA Endevor SCM Element

This is a two-step process. The JCL in member BC1JFUP1, found in the iprfx.iqual.CSIQJCL library, allows you to generate the following:

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

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

- The DDname BSTIPT01 is where the SCL is specified.
- The DDname C1PRINT specifies the output file created by Step 1. If a level is not specified, then the current level will be printed. The DCB attributes for this file are LRECL=133, RECFM=FB.
- The DDname C1CHGSI specifies the temporary file created by Step 1 as shown on the DDname C1PRINT.
- Output created by the IEBUPDTE Request Card Generator is written to the DDname C1UPDTO. The DCB attributes associated with this file are LRECL=80, RECFM=FB.
- The PARM accepted as input by the program BC1PFUDP specifies the position of the sequence number within the element. Two parameters can specify the beginning position and length of the sequence numbers. PARM rules are as follows:
  - The first parameter, as shown in the JCL, is SEQBEG=. The default for SEQBEG is 73.
  - The second parameter is SEQLNG=. The default for SEQLNG is 7. Valid range for SEQLNG is 1-8.
  - If SEQBEG is coded, than SEQLNG must be specified.
  - The NAME= parameter must not be specified.

How to Generate Control Cards When Two Members Differ

This utility operates independently of CA Endevor SCM.

This is also a two-step process. The JCL in member BC1JFUP2, found in the iprfx.iqual.CSIQJCL library, you to generate the following:

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

Certain statements within the sample JCL control the generation of IEBUPDTE control cards: These are as follows:

- The DDname NDVRIPT is where the compare columns are specified. The parameters OUTPUT CHANGES and FORMAT IEBUPDTE must be specified as shown in the example.
- The DDnames NDVRIN1 and NDVRIN2 specify the two files to be compared. (Refer to the utility BSTPCOMP for more information.)
- The DDname NDVRPCH specifies the output file created by Step 1. The DCB attributes for this file are LRECL=88, RECFM=FB.
- The DDname C1CHGSI specifies the temporary file created by Step 1 as shown on the DDname C1PRINT.
- DDname C1UPDTO. The DCB attributes associated with this file are LRECL=80, RECFM=FB.
- The PARM accepted as input by the program BC1PFUDP specifies the position of the sequence number within the element, and the member name associated with the ./ UPDATE card. Three parameters can be specified. PARM rules are as follows:
  - The first parameter, as shown in the JCL, is SEQBEG=. The default for SEQBEG is 73 .
  - The second parameter is SEQLNG=. The default for SEQLNG is 7. Valid range for SEQLNG is 1-8.

- If SEQBEG is coded, than SEQLNG must be specified.
- The third parameter is NAME=. This parameter controls the member name generated on the ./ UPDATE control card. This parameter is required.

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

4

3000

9

3001

10

3002

11

3003

12

3005

13

3006

0

3007

Chapter 5: Using The CONCALL Utility

This section contains the following topics:

The CONCALL Utility

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

Using CONCALL, you can invoke your programs from non-authorized libraries. Generally, programs that are invoked directly from NDVRC1 must reside in an authorized library. This utility enables you to bypass the STEPLIB (or LINKLST) residency requirement for the program specified in the execution parameter. CONCALL can be used to invoke any program from a specified library. CONCALL can also be used be used in conjunction with the NDVRC1 server program to invoke batch programs from a non-authorized library.

Example: CONCALL Utility

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

Chapter 6: Using the Expand Includes Utility

This section contains the following topics:

The Expand Includes Utility

Note: The Expand Include utility program is functionally stable. CA Support will answer questions as well as diagnose and correct defects, in accordance with the CA Technologies Support Policy and Terms. However, we will not provide any additional modifications that alter the syntax logic to accommodate specific customers' implementations. For more information, see TEC536992 "Functional stabilization of the CA Endevor ® SCM “Expand Include” utility program (ENBX1000)" on Support.ca.com.

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

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

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

If you use CA Librarian to expand your COBOL COPY statements, you may use the Expand Includes utility to expand the COPY statements. The Expand Includes utility also supports the REPLACING/BY keyword. For more information, see Processing Methods for Replacing Phrases (see page 82)

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

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

The Expand Includes utility performs the following steps:

1. The utility reads the source file.
2. For each source record:
   a. The utility searches for ++INCLUDE, -INC, or COPY statements.
   b. If an ++INCLUDE, -INC, or COPY statement is found, the utility searches the ENXINCnn libraries for a member name.
   c. If a member name is found, the utility incorporates the source file into the output file. For more detail, see COPY Statement Examples in this chapter.

   After reading all source records, the utility stops processing when the end of the file is reached.

Example: Using the Expand Includes Utility

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

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

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

COPY Statement Examples

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

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

If the original text contains:

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

the output would contain:

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

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

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

If the original text contains:

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

the output would contain:

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

Similarly, suppose you want to replace the original text with this string:

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

If the original text contains:

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

the output would contain:

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

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

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

If the input contains:

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

The output would contain:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
PROCEDURE-LABEL. COPY MNO.
```

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

```
PROCEDURE-LABEL.
```

Expand Includes Utility Processing Modes

The Expand Includes utility executes in one of two modes:

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

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

Expand Includes Utility Input and Output Data Sets

The attributes of the source and destination data sets used in the Expand Includes utility are as follows:

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

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

Processing Methods for Replacing Phrases

The Expand and Replace utility can use the following methods to process the REPLACING phrases on LIBRARIAN -INC or COBOL COPY statements:

- Librarian CCOPY—Using this method, the expansion is processed as follows:
  - Each operand of the Replacing phrase is considered a word.
  - When scanning for a match, the matching string must be separated from other strings with spaces.
  - When strings are enclosed in quotes, the quotes are are included in the scan.
- VSCOBOL COPY—Using this method, the expansion is processed as follows:
  - Each operand of the replacing phrase is considered a string.
  - When scanning for a match, any matching string is processed.
  - When strings are enclosed in quotes, the quotes are removed before scanning.
- COBOL II COPY—Using this method, the expansion is processed as follows:
  - The expansion is processed the same as the COBOL II compiler processes COBOL II COPY replacing syntax.

Select Processing Method for Replacing Phrases

You can specify which processing method the Expand and Replace utility is to use to process the REPLACING phrases on LIBRARIAN -INC or COBOL COPY statements. Select one of the following expansion methods for the REPLACING phrases:

- Librarian CCOPY

  This is the default, but it is also selected by including a EN$CCOPY DD statement in the execution JCL.

- VSCOBOL COPY

  To select this method, include a EN$COBOL DD statement in the execution JCL.

- COBOL II COPY

  To select this method, enable the option COPY_COBOL2_REPLACE on in the ENCOPTBL options table.

Expand Includes Utility Operating Considerations

Consider the following when using the Expand Includes utility:

- Checking the CA Endevor SCM Defaults Table - During initialization, the Expand Includes utility checks the CA Endevor SCM Defaults Table to determine whether CA Librarian or CA Panvalet support is active. The LIBENV= parameter indicates whether support is active and, if so, for which application. If neither CA Librarian nor CA Panvalet is active, the utility issues an error message and terminates immediately.
- Embedding and Looping INCLUDE Statements - The Expand Includes utility expands embedded INCLUDE statements. An embedded INCLUDE statement occurs when a member expanded by an ++INCLUDE, -INC, or COPY statement contains another ++INCLUDE, -INC, or COPY statement.

  The utility also detects looping INCLUDE statements. A looping INCLUDE statement occurs when one member includes another member which, in turn, includes the first member. In this situation, the utility issues an error message and immediately stops processing.
- Superset Support - The Expand Includes utility provides support for CA Panvalet Supersets as follows:

  1. If the INCLUDE Library is a CA Panvalet data set, the utility provides full superset support.
  2. If the INCLUDE library is a partitioned data set, the utility provides limited support. The utility ignores the superset name and expands only the member name. For example, assume the source program contains the following statement:

  ```
  ++INCLUDE superset.member1
  ```

  The Expand Includes utility expands the statement only by looking for member1 in the INCLUDE libraries.
- Security - The Expand Includes utility does not perform any security checking. The utility relies on your site's system (RACF, CA Top Secret, CA ACF2) to enforce data set access.
- Monitoring Components in the Expand Includes Utility - When using the Expand Includes utility in a processor, both CA Librarian and CA Panvalet components have the ability to collect component data for expand include utility or CONWRITE. Component monitoring is also available using CONWRITE with PARM='EXPINCL(y)'.

How the Expand Includes Utility Identifies the INCLUDE Member

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

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

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

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

INCLUDE Statement Source File Format

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

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

How the Expand Includes Utility Works with CA Panvalet Files

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

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

How the Expand Includes Utility Works with CA Librarian Files

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

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

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

How the Expand Includes Utility Works with COBOL COPY Statements

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

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

Note: For examples of rules and formats, see COPY Statement Examples (see page 79).

How the Expand Includes Utility Specifies INCLUDE Libraries

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

The libraries are identified by the following JCL statement:

```
ENXINCnn DD
```

nn can be a two digit number between 00-99 inclusive, or two alphabetic characters in the following series:

- AA-AZ
- BA-BZ
- CA-CZ
- DA-DZ

To resolve ++INCLUDE, -INC, or COPY statements, the utility searches up to 204 INCLUDE libraries.100 in numerical sequence followed by 104 in alphabetical sequence.

Note: In the ENXINCnn DD statement, the execution JCL must include at least one valid ENXINCnn DD statement. In addition, the ENXINCnn DD statement cannot specify a concatenated data set. Finally, the ENXINCnn DD statements can contain a combination of partitioned and CA Panvalet or CA Librarian data sets.

How the INCLUDE Library is Searched

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

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

Partitioned Data Sets

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

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

The Default Location Processing Mode

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

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

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

Default Location mode is the default processing mode.

Example: Default Location Processing Mode JCL

The JCL that executes the Expand Includes utility in Default Location mode can be found in member ENBXDLM1, in the JCL library iprfx.iqual.CSIQJCL.

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

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

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

The ENXIN and ENXOUT DD Statements

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

If	Then
ENXIN DD is a sequential data set	ENXOUT DD must be sequential or partitioned data set with an explicit member name.
ENXIN DD is a sequential data set and ENXOUT DD is a partitioned data set without an explicit member name	You receive an error message.
ENXIN DD is a partitioned data set and the member is identified in the PARM= statement or in the ENXIN DD statement	ENXOUT DD can be either a sequential or partitioned data set.

If	Then
ENXOUT DD is a partitioned data set and an explicit member name is not provided on that DD statement	The utility creates a member with the same name as the input member. The utility always replaces the destination member.
ENXIN DD is a partitioned data set with no member specified, and no member is named in the PARM= statement	You receive an error message.
ENXIN DD is a partitioned data set with an explicit member name and the PARM= statement contains a member name	The PARM= member name is ignored.

The Control Statement Mode

Control Statement mode is activated only if the ENXSCLIN DD statement is allocated in the execution JCL. The Expand Includes SCL statement identifies the source and destination files that will be processed.

Note: For more information about the Expand Includes SCL, see Expand Includes SCL (see page 90).

The JCL that executes the Expand Includes utility in Control Statement mode can be found in member ENBXCSM1, in the JCL library iprfx.iqual.CSIQJCL.

How the Control Statement Mode Processes Members

Control statement mode allows you to process multiple members, or files, in a single execution. You can do this in one of three ways:

- Specify multiple EXPAND INCLUDES actions in the ENXSCLIN DD statement.
- Specify a wildcarded member name in the FROM DSNAME MEMBER clause in the EXPAND INCLUDES action.
- Use a combination of the previous two methods.

If the execution JCL contains a PARM= statement that specifies a member name, in addition to the control statements in the ENXSCLIN DD statement, the utility ignores the member specified on the PARM= statement and issues a warning message.

How Expand Includes Input SCL is Validated

CA Endevor SCM validates the input SCL (EXPAND INCLUDES actions). If no errors are detected, all statements are processed and the Expand Includes Execution Report and Expand Includes Summary Report are produced.

If errors are found in the SCL, CA Endevor SCM continues parsing the statements, but does not process them. Results of the validation process are presented on the Expand Includes Control Statement Summary Report.

The JCL Parameter

The JCL PARM= statement is used for two purposes:

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

You are required to code this parameter.

The PARM= Parameter

The PARM= parameter appears as follows in the JCL:

```
PARM='ENBX1000member'
```

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

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

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

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

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

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

The Member Name

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

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

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

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

Expand Includes SCL

This section explains the EXPAND INCLUDES statement that is used to process a member from a partitioned data set.

You can enter as many EXPAND INCLUDES statements as necessary. These statements are specified in the ENXSCLIN DD statement.

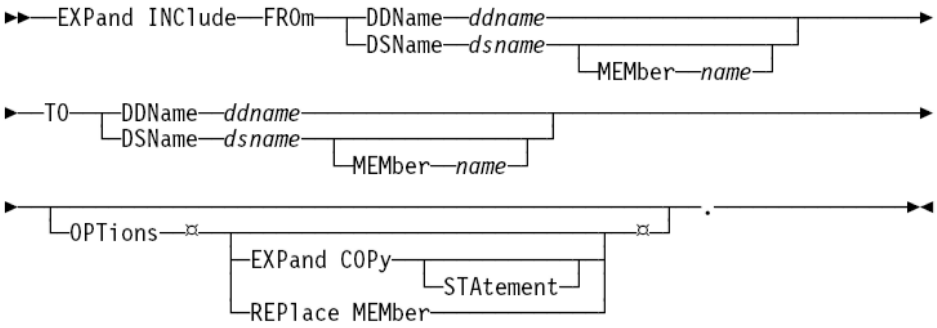
The Expand Includes utility parses and validates all requests before it begins executing them. If there is a syntax error in any request or an error is found validating a request, none of the statements are executed. The utility attempts to parse all of the control statements before terminating, however.

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

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

Expand Includes Syntax

The Expand Includes syntax is shown as follows.



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

Note: For more information about syntax conventions, see the *SCL Reference Guide*.

The EXPAND INCLUDES Clause

The EXPAND INCLUDES clause is the first clause in the statement. The following describes this clause:

EXPAND INCLUDES

The name of the action. You must code this clause.

The FROM Clause

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

FROM DDNAME ddname

Identifies the source data set by DD name. Specify the name of a preallocated DD statement.

FROM DSNAME dsname

Identifies the source data set by data set name. Specify the name of an existing data set, using standard CA Endevor SCM naming conventions. If the data set name contains embedded periods, enclose the name in quotation marks.

The data set referred to must be either sequential or partitioned. The data set record format can be either fixed or variable.

MEMBER name

Identifies the member(s) to be processed from a partitioned data set. This clause is required.

The member name must meet the following specifications:

- The member name can be no longer than eight characters. You can use a wildcard.
- The MEMBER clause applies only if the input data set specified in the FROM DSNAME clause is a partitioned data set. If the data set specified is sequential, the MEMBER clause is ignored and a warning message issued.
- If the member name is fully specified, the member must exist in the input data set. If the member name is wildcarded, at least one member matching the wildcard criteria must exist in the input data set. If the explicit member does not exist, or no matches can be found, you will receive an error message.

The TO Clause

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

TO DDNAME ddname

Identifies the destination data set by DD name. Specify the name of a preallocated DD statement.

TO DSNAME dsname

Identifies the destination data set by data set name. Specify the name of an existing data set, using standard CA Endevor SCM naming conventions. If the data set name contains embedded periods, enclose the name in quotation marks.

The data set referred to must be either sequential or partitioned. The data set record format can be either fixed or variable. The record length must be at least as long as the record length of the input data set and at least as large as the INCLUDE libraries associated with the ENXINCnn DD statements. If the data set record length is not large enough for either condition, the Expand Includes utility truncates the output records and issues a caution message.

MEMBER name

Identifies the name of the output member. This clause is optional.

The member name must meet the following specifications:

- The member name can be no longer than eight characters and cannot be wildcarded.
- The MEMBER clause applies only if the output data set specified in the TO DSNAME clause is a partitioned data set and if the input data set is either a sequential data set or a partitioned data set with an explicit member name (that is, not wildcarded). If the input data set is a partitioned data set and the MEMBER clause contains a wildcarded member name, you will receive an error message.
- If the FROM DSNAME MEMBER clause contains a wildcarded member name or if the FROM DSNAME is a partitioned data set and the FROM MEMBER clause is not specified, the TO MEMBER clause cannot be specified. You cannot rename multiple output members. In this situation, you will receive an error message.

The OPTIONS Clauses

The Expand Includes syntax contains two optional clauses:

EXPAND COPY [STATEMENTS]

Tells the Expand Includes utility to expand COBOL COPY statements. This clause is an alternative to coding the EXPANDCOPY parameter in the JCL.

If you do not code this clause but do include the EXPANDCOPY parameter on the JCL PARM= statement, the utility will expand the COBOL COPY statements.

REPLACE MEMBER

Tells the Expand Includes utility to replace an existing member in the output data set. This clause applies only to partitioned data sets. If the destination data set is a sequential data set, this clause is ignored.

If this clause is not specified and the member being created currently exists in the output data set, the EXPAND INCLUDES action fails for that member. Processing continues for other members associated with the request.

Expand Includes Utility Reports

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

Control Statement Summary

Shows the control statements that were provided in the ENXSCLIN DD statement and identifies any parser or statement validation errors.

Execution Report

Contains information about the execution of each request.

- If the utility is executing in Default Location mode, the report contains information about the single request.
- If the utility is executing in Control Statement mode, the utility generates detailed information about each EXPAND INCLUDES request.

Expand Includes Summary Report

Summarizes each request processed. The summary indicates the member name, the return code, the number of INCLUDE members, and the number of lines expanded.

These reports are written to the ENXMSGS1 DD statement. If the execution JCL includes an ENXMSGS2 DD statement, the utility writes the Expand Includes Summary Report to that file.

The Expand Includes Control Statement Summary Report

The Expand Includes Control Statement Summary shows the control statements provided in the ENXSCLIN DD statement, and whether there are any parser or validation errors. This report is generated only if the program is executing in Control Statement mode.

The Expand Includes Execution Report

The Expand Includes Execution Report contains execution information for each request.

The Expand Includes Summary Report

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

Statement Number

The statement number associated with the EXPAND INCLUDES action. If the utility is running in Default Location mode, the statement number is always 1.

Data Set Name

The name of the input data set.

Member Name

The name of the input member. This field is blank if the input file is a sequential data set.

Return Code

The return code associated with the EXPAND INCLUDES request for the data set or member.

Number of Include Members

The number of ++INCLUDE, -INC, or COPY members that were expanded.

Number of Lines Expanded

The number of lines added to the output file from the expanded ++INCLUDE, -INC, or COPY members.

Chapter 7: Using the Library Conversion Utilities

This section contains the following topics:

The Library Management Conversion Process

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

There are three phases to the library management conversion process:

Analyze (CA Endevor SCM Inventory Analyzer)

Analyzes the members in your CA Panvalet or CA Librarian files, creating load SCL for input into CA Endevor SCM. It is assumed that you have already run the CA Endevor SCM Inventory Analyzer against your inventory, and that you have identified the CA Endevor SCM inventory structure--that is, environments, systems, subsystems, and, optionally, processor groups--that will be assigned to each element.

Note: For detailed information about the analysis process, see the *Inventory Analyzer Guide*.

Load (CA Endevor SCM Load utility)

Loads appropriate members into CA Endevor SCM.

Validate (Member Validation Program)

Validates that all members in a single data set exist in a specified CA Endevor SCM environment.

CA Panvalet Libraries

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

Supersets (Panvalet Only)

Supersets apply to CA Panvalet only.

The analysis phase of the conversion process produces a reference data set that contains a list of members. These members have been analyzed for specific information; analysis is done alphabetically. During this phase, the conversion process identifies members that are supersets as well as members that reference supersets. Appropriate messages are returned when the list is generated, as shown in the following table:

Member	Message
Is a superset	CAE$0003 MEMBER member name IS A SUPERSET
Is not a superset	No message
References a member that has already been analyzed and is a superset	CAE$0004 MEMBER member name REFERENCES SUPERSET superset name
References a member that is a superset but has not yet been analyzed	No message

Example: Handling Supersets

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

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

Member	Is a superset?	References a superset?	Issues this message
.	.	.	.
.	.	.	.
Z	Yes	No	CAE$0003

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

The Analyze Phase

This section describes the first phase of the conversion process that analyzes the members in your inventory to identify the following:

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

This analysis is performed by the CA Endevor SCM Inventory Analyzer, which is invoked by the conversion job stream. The conversion job stream consists of four steps, briefly described as follows:

1. Delete SCL output data sets. In this step, existing output data sets are deleted so new ones can be created.
2. Build reference data set. In this step, a list of analyzed members is created that will be used as input in the next step.
3. Identify INCLUDE and COPY members, and build SCL. In this step, INCLUDE and COPY members are identified using the information generated in the previous step. Load SCL is created for input into CA Endevor SCM.
4. Identify superset members. In this step, superset member and unresolved members (members referenced but not found) are identified.

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

The Conversion Job Stream

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

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

Important! Throughout the conversion job stream, there are several statements that begin with the words ESTABLISH TYPE. These statements are the rules used by the CA Endevor SCM Inventory Analyzer to identify members that contain INCLUDE or COPY statements and to classify the members with a CA Endevor SCM type. These rules are the only rules needed for the conversion process. Do not change these rules.

Element Classification

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

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

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

- The element name is the same as the library member name.

- CA Endevor SCM elements require a type. The conversion job stream identifies the element type to be assigned to the member.

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

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

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

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

    ```
    "I  "
    ```

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

    ```
    "   "
    ```

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

    The conversion job stream uses the CA Panvalet comment or CA Librarian description as the CA Endevor SCM comment.

The PROC Definition

This section describes the PROC definition portion of conversion job stream and the variables for which you need to provide values before submitting the job.

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

```
// (JOBCARD)
//*
//*********************************************************************
//*  THIS JCL MUST BE TAILORED PRIOR TO SUBMITTING THIS JOB     *
//*                                  *
//*   1. CHANGE TDISK TO A UNIT NAME FOR WORK DASD ——       *
//*     FOR BEST PERFORMANCE, SPECIFY A VIO DEVICE        *
//*                                  *
//*     CHANGE iprfx.iqual TO YOUR CA Endevor SCM INSTALL PREFIX   *
//*                                  *
//*                                  *
//*   2. CHANGE PDISK TO A UNIT NAME FOR DATA SETS WHICH     *
//*     WILL CONTAIN CA Endevor SCM LOAD SCL STATEMENTS USED FOR   *
//*     INPUT INTO THE CA Endevor SCM LOAD UTILITY JOB        *
//*                                  *
//*     CHANGE PVOLSER (OR REMOVE VOL=SER=PVOLSER) TO A VALID  *
//*     VOLUME SERIAL NUMBER                  *
//*                                  *
//*     CHANGE UPRFX.UQUAL TO THE APPROPRIATE INDEX LEVELS    *
//*     AT YOUR SITE                        *
//*                                  *
//*   3. TAILOR THE THREE LINES THAT CONTAIN THE TEXT       *
//*     'DSN1' 'DSN2' 'DSN3' TO REFLECT THE ACTUAL LIBRARY    *
//*      NAMES FROM WHICH YOU ARE CONVERTING.          *
//*                                  *
//*                                  *
//*   4. ON THE ASSIGN STATEMENT USED IN STEP TWO, SPECIFY    *
//*     THE ACTUAL ENVIRONMENT, SYSTEM, SUBSYSTEM, STAGE ID   *
//*     AND CCID VALUES THAT ARE TO BE USED TO CONSTRUCT THE   *
//*     LOAD UTILITY SCL.                    *
//*                                  *
```

```
//*******************************************************************
//ANALYZE PROC
//*
//C1BM7000 EXEC PGM=NBURC1,PARM=C1BM7000
//     DYNAMNBR=1500,REGION=4096K
//*
//CONLIB  DD DSN=iprfx.iqual.CSIQLOAD,DISP=SHR
//*
//C1TPDD01 DD UNIT=tdisk,SPACE=(CYL,3),
//      DCB=(RECFM=VB,LRECL=260,BLKSIZE=6160)
//C1TPDD02 DD UNIT=tdisk,SPACE=(CYL,5),
//      DCB=(RECFM=VB,LRECL=260,BLKSIZE=6160)
//C1TPLSIN DD UNIT=tdisk,SPACE=(CYL,3),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//C1TPLSOU DD UNIT=tdisk,SPACE=(CYL,5)
//C1PLMSGS DD SYSOUT=*
//*******************************************************************
//* OUTPUT DATA SETS                          *
//*******************************************************************
//C1MSGS1 DD SYSOUT=*
//C1SUMARY DD SYSOUT=*
//C1PRINT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6171)
//SYSABEND DD SYSOUT=*
//SYSOUT  DD SYSOUT=*
//BSTERR  DD SYSOUT=*
//     PEND
```

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

iprfx

Highest-level qualifier used to assign data set names for installation files at your site.

iqual

Second-level qualifier used to assign data set names for installation files at your site.

tdisk

Unit name for temporary disk data sets.

How to Delete Output Data Sets

This section describes Step 1 of the conversion job stream that deletes the existing output data sets.

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

```
//STEP1  EXEC PGM=IDCAMS

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

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

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

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

Tailor the JCL by providing values for the following variables:

uprfx

Highest-level qualifier used to assign data set names for CA Endevor SCM user files at your site.

uqual

Second-level qualifier used to assign data set names for CA Endevor SCM user files at your site.

How to Build Reference Data Set

This section describes Step 2 of the conversion job stream that creates a reference data set by analyzing members in specified data sets.

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

```
//STEP2  EXEC ANALYZE
//*********************************************************************
//* BUILD REFERENCE DATASET                                           *
//*********************************************************************
//BSTPUNCH DD DSN=&&BSTPUNCH.,DISP=(NEW,PASS,DELETE),
//      UNIT=tdisk,SPACE=(TRK,(20,10),RLSE),
//      DCB=(RECFM=FB,LRECL=200,BLKSIZE=22000)
//BSTIPT01 DD *
 ANALYZE MEMBER *
   FROM DSNAME 'DSN1' 'DSN2' 'DSN3'
 .
//BSTRULES DD *
ESTABLISH TYPE INCLUDE GROUP INCLUDE WHEN
 LIB = INCLUDE OR PAN = INCLUDE.
ESTABLISH TYPE INCLUDE GROUP COPY WHEN
 LIB = COPY.
ESTABLISH TYPE REMAINDR GROUP THATSALL WHEN
 PAN = '' AND LIB = ''.
DEFINE PAN INCLUDE WHEN
 '&C1FDSN$IO'. = 'PAN'
AND
 SOURCE TEXT CONTAINS
 '++INCLUDE' IN COLUMN 8 INVOKE EXIT=C1BM7CAE
 .
DEFINE LIB INCLUDE WHEN
 '&C1FDSN$IO'. = 'LIB'
```

```
AND
 SOURCE TEXT CONTAINS
 '-INC' IN COLUMN 1 INVOKE EXIT=C1BM7CAE
 .
DEFINE LIB COPY WHEN
 '&C1FDSN$IO'. = 'LIB'
AND
 '&C1FDSN$LANG'. = 'CBL'
AND
 SOURCE TEXT CONTAINS
 ' COPY ' IN COLUMNS 7 THROUGH 68
     INVOKE EXIT=C1BM7CAE WITHOUT
     '*' IN COLUMN 7 LINE CURRENT
 .
//*
//*
```

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

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

Tailor the JCL by providing values for the following variables:

tdisk

Unit name for temporary disk data sets.

DSN1, DSN: hp1.2,...DSNn

The name(s) of the data sets you want scanned. You can code up to 16 names per ANALYZE statement.

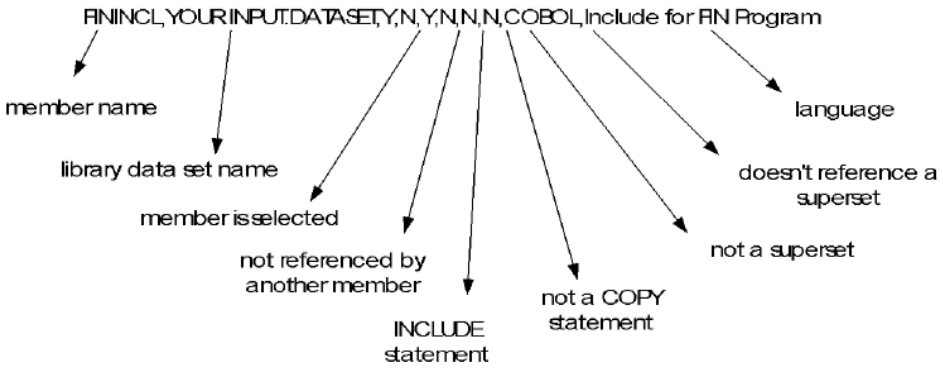
Two types of records are created in this step: a header record and detail records. The header record assigns symbolic names to the data in the detail records. Each detail record contains the information listed as follows, in the order shown, for each unique occurrence of the member:

- Member name
- Library data set name
- Yes and no (Y/N) indicators for the following:
  - Was the member selected for processing?
  - Was the member referenced by another member?
  - Was the member referenced as an INCLUDE statement?

  - Was the member referenced as a COPY statement?
  - Is the member a superset?
  - Does the member reference a known superset?
- Language
- Comment

Example: A Typical Record in the Reference Data Set

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



How to Build Load SCL

This section describes Step 3 of the conversion job stream that creates the Load SCL.

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

```
//STEP3  EXEC ANALYZE

//*********************************************************************
//* IDENTIFY INCLUDE AND COPY MEMBERS, BUILD LOAD SCL                 *
//*********************************************************************
//BSTPUNCH DD DSN=&&BSTPUNCH.,DISP=(OLD,PASS)
//INCLMBRS DD DSN=uprfx.uqual.INCL.SCLSTMTS,DISP=(,CATLG),
//      UNIT=pdisk,vol=ser=pvolser,SPACE=(TRK,(3,2),RLSE),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160,DSORG=PS)
//PGMMBRS DD DSN=uprfx.uqual.NONINCL.SCLSTMTS,DISP=(,CATLG),
//      UNIT=pdisk,vol=ser=pvolser,SPACE=(TRK,(3,2),RLSE),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160,DSORG=PS)
//BSTIPT01 DD *
*
 SET ASSIGN
  ENV = 'ENVNAME'
  SYS = 'SYSNAME'
  SBS = 'SBSNAME'
  CCID = 'CCID' .
*
 SET REFERENCE DDNAME BSTPUNCH.
*
 ANALYZE MEMBER *
   FROM DSNAME 'DSN1' 'DSN2' 'DSN3'
 .
*
//BSTRULES DD *
DEFINE INCLUDE SUPERSET WHEN
 '&CAESUPER'. EQ 'Y'
 .
ESTABLISH TYPE COPY GROUP NOPROC WHEN
 '&CAECOPY'. = 'Y' AND INCLUDE NOT = SUPERSET
 .
ESTABLISH TYPE INCLUDE GROUP NOPROC WHEN
 '&CAEINC'. = 'Y' AND INCLUDE NOT = SUPERSET
 .
```

```
//BSTMODEL DD *
%DDNAME=INCLMBRS,COND=SUCCESS
 LOAD MEMBER &C1MEMBER
  FROM DSNAME '&C1FDSN'.
  TO ENVIRONMENT &ENV.
    SYSTEM    &SYS.
    SUBSYSTEM  &SBS.
    TYPE     &C1TYPE(1.,1)&CAELANG(1.,7)
  OPTIONS CCID '&CCID'.
    COMMENT '&CAEDESC'.
 .
%DDNAME=PGMMBRS,COND=FAILURE
 LOAD MEMBER &C1MEMBER
  FROM DSNAME '&C1FDSN'.
  TO ENVIRONMENT &ENV.
    SYSTEM    &SYS.
    SUBSYSTEM  &SBS.
    TYPE     &CAELANG(1.,8)
  OPTIONS CCID '&CCID'.
    COMMENT '&CAEDESC'.
```

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

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

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

Tailor the JCL by providing values for the following variables:

uprfx

Highest-level qualifier used to assign data set names for CA Endevor SCM user file at your site.

uqual

Second-level qualifier used to assign data set names for CA Endevor SCM user files at your site.

pdisk

Unit name for permanent disk data sets. These data sets will contain CA Endevor SCM Load SCL statements.

pvolser

Volume serial number of the disk. Either enter a valid volume serial number or delete the parameter VOL=SER=PVOLSER.

ENVNAME

Environment name.

SYSNAME

System name.

SBSNAME

Subsystem name.

CCID

CCID to be associated with the element.

DSN1, DSN2,...DSN*n*

The name(s) of the data sets you want scanned. Use the same data set names you coded in Step 2.

Note: You *must* enter the data set names in the same sequence as they were entered in Step 2.

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

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

Load Syntax Variables

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

&C1MEMBER (Member name)

Reference data set

&C1FDSN (FROM data set name)

Reference data set

&ENV (Environment name)

SET ASSIGN parameter

&SYS (System name)

SET ASSIGN parameter

&SBS (Subsystem name)

SET ASSIGN parameter

&C1TYPE (1,1) INCLUDE (I) or COPY (c)

Reference data set.

&CAELANG (1,7)-CA Endevor SCM type

CA Panvalet or CA Librarian language directory

&CCID (CCID associated with the member)

SET ASSIGN parameter

&CAEDESC (Comment associated with the member)

CA Panvalet comment or CA Librarian description

&CAELANG (1,8)-CA Endevor SCM type

Reference data set (if not an INCLUDE or COPY member)

How to Identify Superset Members

This section describes Step 4 of the conversion job stream that identifies the superset members.

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

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

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

For this step, you need only code the data set names you used in Step 2 and Step 3. You *must* enter the data set names in the same sequence as entered in the previous steps. For unresolved members, CA Endevor SCM writes messages to the Execution Log, identifying the member by member name.

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

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

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

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

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

The Load Phase

The second phase of the conversion process loads the members classified in the first phase into CA Endevor SCM.

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

The Load Utility

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

Note: The Load utility JCL is provided on the installation tape, in member BC1JLOAD of the JCLLIB.

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

Provide values for the following variables:

iprfx

Highest-level qualifier used to assign data set names for installation files at your site.

iqual

Second-level qualifier used to assign data set names for installation files at your site.

uprfx.uqual.INCL.SCLSTMTS

The name of the data set that contains the load statements (output from phase 1 of the conversion process).

The Load Utility Output

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

The following describes the report name and when the report is produced:

CA Endevor SCM LOAD Execution Log

Always

CA Endevor SCM Data Validation Report

Only when the requests contain invalid data

CA Endevor SCM LOAD Execution Report

Only when the requests contain no syntax errors or invalid data

CA Endevor SCM LOAD Execution Summary

Only when the requests contain no syntax errors or invalid data

The Validate Phase

The third phase of the conversion process checks if the members have been loaded correctly loaded into the CA Endevor SCM.

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

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

The Member Validation Program

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

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

Return Codes (Member Validation Program)

The Member Validation Program passes the following return codes:

0

All members in the source data set were found in the CA Endevor SCM environment specified.

4

One or more members were not found in the environment specified.

12

The Member Validation Program encountered an error.

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

The Member Validation Program JCL

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

'ENBRVDSN *environment_name*'

The CA Endevor SCM environment that will be searched for members. If you do not code this parameter, or the value is invalid, you receive an error message.

Specify an up to eight character, valid CA Endevor SCM environment name.

Note: There are no characters between ENBRVDSN and *environment_name*.

CONLIB DD

Standard JCL statement.

Change IPRFX and IQUAL to the qualifiers you are using for your site.

ENVMSGS1

The destination of the Member Validation Report. The DD statement usually allocates a SYSOUT data set.

No tailoring required.

ENVDSN00

The data set that is to be validated.

Specify the name of a data set (*source.dataset.name*) generated by the conversion job stream. The data set must be partitioned, CA Panvalet, or CA Librarian.

The Member Validation Report

This section describes the Member Validation Report that is generated by the Member Validation Program. This report contains information about the members loaded into the CA Endevor SCM.

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

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

The report is written to the ENVMSGS1 DD statement.

Multiple Occurrences of the Member

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

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

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

Chapter 8: Using the Load Utility

This section contains the following topics:

The Load Utility

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

Security is invoked when the Load Utility is executed. If using ESI, a PRIMARY_OPTIONS call is issued. For this ESI call, the MENUITEM is LOAD.

Before using the CA Endevor SCM Load Utility, the inventory structures that you want to populate within CA Endevor SCM must be defined.

Note: For more information about defining inventory structures, see the *Administration Guide*.

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

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

The CA Endevor SCM Load Utility is simple to operate and involves two basic steps:

Creating your requests

The Load Utility automatically validates each request before it is executed.

Reviewing the reports produced

Reports are produced during and after execution to inform you about what has occurred.

How to Create LOAD Requests

LOAD requests indicate those members, from designated data sets, that are to be loaded to specific CA Endevor SCM locations. A sample request is shown as follows:

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

This request tells the Load Utility to perform the following actions:

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

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

LOAD Request Reports

As request execution progresses, the following reports are produced:

- The CA Endevor SCM Load Execution Log
- The CA Endevor SCM Data Validation Report
- The CA Endevor SCM Load Execution Report
- The CA Endevor SCM Load Summary Report

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

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

CA Endevor SCM Load Utility Requests

This section describes the syntax you use to load elements into environments pre-defined in CA Endevor SCM. The Load process is described in the next section of this chapter. That discussion includes syntax examples as well as explanations and illustrations of the reports you can use to review what you have coded.

Load Utility Statements

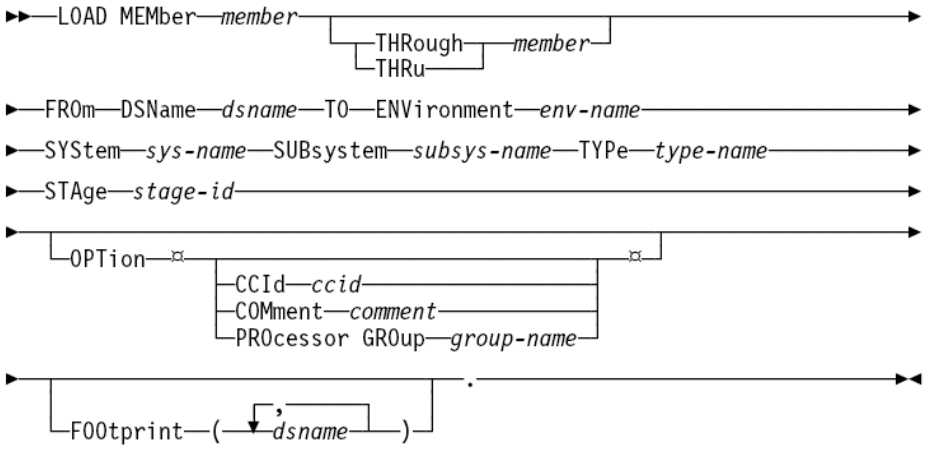
The Load Utility uses three types of statements:

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

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

Load Request Syntax

The following is the syntax for the load request:



Load Request Rules

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

Load Request Required Clauses

Required clauses are listed as follows:

LOAD MEMBER *member*

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

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

FROM DSNAME *dsname*

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

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

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

```
'TEST.LIB' or "TEST.LIB"
```

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

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

Indicates the CA Endevor SCM location to which the member(s) will be loaded. If you do not provide (all) TO information here, a SET TO clause with the required data must have been previously coded.

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

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

Load Request Optional Clauses

Optional clauses are listed as follows:

THROUGH (THRU) *member*

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

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

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

OPTIONS

CCID ccid

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

COMMENT comment

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

PROCESSOR GROUP group-name

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

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

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

FOOTPRINT ('dsname', 'dsname')

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

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

- When the Load Utility footprints an object deck, the content and directory of the library are both footprinted. Any existing footprints are replaced.
- When the Load Utility footprints a load module, a composite footprint is created. That is, the entire load module, as opposed to just individual parts, is footprinted. Footprints currently existing in the load module are not affected by the Load Utility's footprint. The composite footprint is placed in the ZAP IDR record for the load module.
- The FOOTPRINT data set(s) can be partitioned data sets (PDS or PDS/E) , or CA Librarian or CA Panvalet libraries. If the data set name(s) includes periods, be sure to enclose the name in single or double quotes.
- You can also specify multiple data sets. When you do so, you must enclose all the data set names in a single set of parentheses, separating them with blanks or commas. Parentheses are not required if you code only one data set name.
- If you use the FOOTPRINT clause, at least one entry must be included. You can code one or more data set names, as described previously, or you can code a clause similar to the following:
- FOOTPRINT ' '.
- This entry is simply a blank(s) enclosed by quotes. CA Endevor SCM interprets this clause as having no data sets designated, which in turn means that no footprinting will occur.
- If you do not code any footprint information in this clause, the Load Utility checks to see whether a SET FOOTPRINT clause has already been specified, and applies that data accordingly.
- If you want to footprint members in the FROM data set, you must include that data set name in the FOOTPRINT clause.

Set Statements

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

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

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

Each SET statement remains in effect until:

- You code specific values in the actual LOAD statement, which override the corresponding values in the SET statement.
- CA Endevor SCM encounters another, like SET statement, which overrides the existing SET statement.
- CA Endevor SCM encounters a CLEAR statement for that particular SET statement.
- Processing for the job ends.

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

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

The SET FROM Statement

The syntax of SET FROM statement is shown as follows:

SET FROM DSNAME *dsname*

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

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

The SET TO Statement

The syntax of the SET TO statement is shown next:

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

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

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

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

The SET OPTIONS Statement

The syntax of the SET OPTIONS statement is shown next:

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

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

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

The SET FOOTPRINT Statement

The syntax of the SET FOOTPRINT statement is shown next:

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

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

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

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

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

```
FOOTPRINT '  ' .
```

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

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

Clear Statements

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

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

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

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

Each of these is discussed briefly as follows.

CLEAR ALL

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

CLEAR FROM

CLEAR FROM clears all SET FROM information previously coded.

CLEAR TO

CLEAR TO clears all SET TO information previously coded.

CLEAR OPTIONS

CLEAR OPTIONS clears all SET OPTIONS information previously coded.

CLEAR FOOTPRINT

CLEAR FOOTPRINT clears all SET FOOTPRINT information previously coded.

Load Utility Reports

The following reports are produced by the CA Endevor SCM Load Utility after execution of the JCL:

- CA Endevor SCM Load Execution Log
- CA Endevor SCM Data Validation Report

- CA Endevor SCM Load Execution Report
- CA Endevor SCM Load Execution Summary

Each of these reports is explained in the following sections.

Note: For examples on their use, see Example of the Load Utility Process (see page 130).

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

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

The reports affected include the following (in assembler):

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

The Load Execution Log

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

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

The Data Validation Report

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

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

The Load Execution Report

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

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

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

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

The Load Execution Summary

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

- The return code for the request.
- The total number of members requested.
- The total number of members in the request that were successfully loaded.
- The total number of members in the request that failed the loading process.

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

It is beneficial to use this report in combination with the CA Endevor SCM Load Execution Report. You can use the Execution Summary to quickly find any requests with errors, then refer back to the more detailed Execution Report to determine where and when the error occurred.

Example of the Load Utility Process

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

How to Load the Request

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

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

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

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

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

How to Execute the JCL

When your requests are ready to be loaded into CA Endevor SCM, submit them for execution, using JCL similar to the following:

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

How to Review the Reports

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

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

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

Report/Condition	Syntax Errors Only	Invalid Data Only	Syntax Errors and Invalid Data	Syntax and Data Correct
CA Endevor SCM Load Execution Log	B	B	B	B
CA Endevor SCM Data Validation Report	Not produced	B	Not produced	Not produced
CA Endevor SCM Load Execution Report	Not produced	Not produced	Not produced	B
CA Endevor SCM Load Execution Summary	Not produced	Not produced	Not produced	B

Load Request Numbers

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

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

The CA Endevor SCM Load Execution Log (DDname = C1BMLLOG)

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

In the following report, the LOAD request was loaded successfully. The LOAD syntax appears first, in the order in which it was coded. Several informational messages follow, one for each member to be added in the LOAD request. Each message indicates that the member requested has been created, and lists the version number and stage associated with the member. The final message, which is the last line of the report, indicates that the load processing for the request was completed successfully.

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

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

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

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

The CA Endevor SCM Data Validation Report (DDname = C1BMLSYN)

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

Assume that an incorrect system name (“BADSYS”) was specified. a CA Endevor SCM Data Validation Report would be produced.

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

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

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

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

The CA Endevor SCM Load Execution Report (DDname = C1BMLDET)

The CA Endevor SCM Load Execution Report is not produced unless every request in the input data set (DDname C1BMLIN) is correct syntactically and contains no invalid data. Remember that the input data set can contain one request with several members, multiple requests of one member each or any combination of the two. If you are loading several (2 or more) members, whether in a single request or using several requests, you may want to review the CA Endevor SCM Load Execution Summary before the CA Endevor SCM Load Execution Report. The CA Endevor SCM Load Summary alerts you to the number of members that failed load processing in a single request, for every request coded. You can then refer back to the CA Endevor SCM Load Execution Report to locate the specific members, in each request, that contain the errors.

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

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

The CA Endevor SCM Load Execution Summary (DDname = C1BMLSUM)

As with the CA Endevor SCM Load Execution Report, the CA Endevor SCM Load Execution Summary is not produced unless every request in the input data set is correct syntactically and contains no invalid data.

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

- **The return code for the request**. In this example, the return code is **0000**, which indicates that processing was successful.
- **The total number of members requested in the LOAD request**. In this example, a range of members was requested, resulting in a total of **5** members to be loaded.

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

The Load Utility Footprint Override Exit

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

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

The Load Utility Footprint Exit Operation

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

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

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

Word 1

Contains the address of the Load Exit Control Block. The control block is mapped by the @LOADDS macro.

Word 2

Contains the address of a 400 byte work area that is available for the exit. The area is on a double-word boundary and is initialized to binary zeroes for each invocation of the exit.

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

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

0

The exit does not want to override the member name.

4

The exit wants to override the member name to be footprinted.

8

The exit encountered an unrecoverable error. The Load Utility will footprint the default member.

The @LOADDS Load Exit Control Block

The following DSECT maps the Load Exit Control Block:

```
LOADDS  DSECT
EXFUNC  DS  H           FUNCTION CODE
EXVER  EQU  1           VERFIY REQUEST
EXFOOT  EQU  2           FOOTPRINT REQUEST
EX$MAXF EQU  2           FUNCTION MAX VALUE
EXSEVI  DS  CL1          MESSAGE SEVERITY SELECTION IND
EXMSGI  EQU  C'I'         INFORMATIONAL MESSAGE SELECTED
EXMSGW  EQU  C'W'         WARNING MESSAGE SELECTED
EXMSGC  EQU  C'C'         CAUTION MESSAGE SELECTED
EXMSGE  EQU  C'E'         SEVERE ERROR MESSAGE SELECTED
EXDDNM  DS  CL8          DDNAME
EXDSNM  DS  CL44         DSNAME
EXDATA  DS  CL65         FOOTPRINT REQUIRED DATA
EXMBRNM DS  CL10         MEMBER NAME
     ORG  EXDATA

EXENV  DS  CL8          ENVIRONMENT
EXSYS  DS  CL8          SYSTEM
EXSBS  DS  CL8          SUBSYSTEM
EXELENM DS  CL10         ELEMENT NAME
EXTYPE  DS  CL8          TYPE
EXSTGN  DS  CL8          STAGE NAME
EXSTG#  DS  CL1          STAGE
EXGRP  DS  CL8          PROCESSOR GRP
EXLVV  DS  CL2          VERSION CHARACTER
EXLVVX  DS  XL1          VERSION (BINARY)
EXLLL  DS  CL2          LEVEL CHARACTER
EXLLLX  DS  XL1          LEVEL (BINARY)
     ORG
EXPRB@  DS  F           ADDRESS OF PRB FOR MESSAGES
EXRESVD DS  4F          ** RESERVED **
EXRQ#LN EQU  *-LOADDS
```

The C1BMLXIT Exit

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

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

The exit can be assembled and link-edited using standard procedures. The load module name that is created must be named C1EXITL1 and it must be placed in the CA Endevor SCM CONLIB library. The program must be link-edited with the RENT, REUS, AMODE(31), and RMODE(24) attributes.

Chapter 9: Using the Notification Utility

This section contains the following topics:

The Notification Utility (BC1PNTFY)

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

- SMTP (email)
- TSO
- TPX
- XMIT

Several sample user exit programs are provided with CA Endevor SCM. The source for these exit programs is located in the iprfx.iqual.CSIQOPTN library.

Note: For more information about these exit programs, see the *Exits Guide*.

How to Configure the Notification Utility

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

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

The following sections explain these parameters and their values.

Universal Parameters

Regardless of which protocol you use to send the notification, the exit program always needs to pass certain parameters to the control block. The following table describes these parameters. Information is given for both assembler and COBOL. Copybooks and Dsects are provided in iprfx.iqual.CSIQOPTN for use in your programs. See members $NOTIFY and NOTIFYDS for assembler and COBOL.

NOTFTYPE (assembler)

NOTI-NOTIFY-TYPE (COBOL)

Specifies the protocol you want to use to send the message. The possible values are:

- SMTP
- TSO
- TPX
- XMIT
- Blank (uses default protocol)

NOTMSGTX (assembler)

NOTI-MESSAGE-TEXT (COBOL)

80-character text field used to specify the message.

Note: If you select a value of **V** for the NOTMSGVF parameter, this field is not used.

NOTFUSER (assembler)

NOTI-USER (COBOL)

The ID of the user or users that will receive the notification. The type of ID depends on the protocol used to send the message.

Protocol-specific Parameters

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

The following sections describe these protocol-specific parameters.

SMTP-specific Parameters

If you are using the SMTP protocol to inform your users and package approvers of a CA Endevor SCM event, use the following information for the SMTP-specific parameters that the exit program must pass to the control block. Information is provided for assembler.

NOTATSIGN

This field allows you to override the default character used to construct email addresses. When no value is entered, CA Endevor SCM uses the default value, which can be either @ sign, or the value coded in $ESMTP table. Specify one byte hexadecimal value, or leave the field blank if you do not want to override the default value. :

NOTMSGVF

Indicates whether the message text format is fixed or variable length. The possible values are:

F-Fixed length. This is the default setting.

V-Variable length.

Note: If you select **F**, or do not select a value, use the NOTMSGTX parameter to specify the message text. Do not use the NOTSMTPM parameter.

NOTSIZE

Indicates the size field in the $NOTIFY control block.

NOTMSGMX

Indicates the maximum message length. Required only if the value of NOTMSGVF is **V**.

NOTSMTPM

The address of the email text. The text must be in the following format:

```
LLLLmessage-text
```

- **LLLL**-The length (in binary) of the message text.
- **message-text**-The message text.

NOTSMTPS

50-character text field used for the email's subject field.

NOTMSGLP

Text field used to specify the message.

- If NOTMSGVF is F, then the data string is treated as a string of fixed records. Fixed records are 84 bytes long, with a 4-byte address of the next record (zero - end), and 80 bytes of text content.
- If NOTMSGVF is V, then the data string is treated as a string of variable records, instead of as a string of a single variable record. The variable record format is a 4-byte address of the next record (zero - end), a 2-byte length of string, and nn bytes of string content (up to 32K bytes).

NOTFROM

A four-byte field that allows you to specify a FROM USER for the emails that are sent. The field should contain the address of an area of up to 50 bytes containing the FROM-ID that is to be used in emails. No spaces are allowed in the FROM-USER field. If the address is zeros, CA Endevor SCM uses a default FROM USER definition.

For a sample exit program that illustrates how to use these parameters, see XIT7MAIL in the iprfx.iqual.CSIQOPTN library.

Note: For more information about XIT7MAIL, see the *Exits Guide*.

TSO-specific Parameters

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

NTSOALLU (assembler)

NOTI-SET-ALL-USER-OPT (COBOL)

Specifies whether you want to send the message to all TSO users. The possible values are:

N-Do not send the message to all TSO users. This is the default setting.

Y-Send the message to all TSO users.

Note: If you select **Y**, the NOTFUSER (assembler) or NOTI-USER (COBOL) field must be left blank.

NTSOLOGN (assembler)

NOTI-SET-LOGON-OPT (COBOL)

Specifies when users receive the message. The possible values are:

Y-Users receive the message when they log on to TSO. This is the default setting.

N-Users do not receive the message when they log on to TSO.

NTSOSAVE (assembler)

NOTI-SET-SAVE-OPT (COBOL)

Specifies whether you want to save the message in a broadcast set. The possible values are:

N-(Default) Do not save the message in a broadcast set.

Y-Save the message in a broadcast set.

TPX-specific Parameters

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

NTPXJOBN (assembler)

NOTI-TPX-JOB-NAME (COBOL)

The site-specific name of the TPX started task. This is a required entry.

NTPXTYPE (assembler)

NOTI-USERID-OPTION (COBOL)

Specifies the type of TPX call to issue. The possible values are:

U-(Default) USERID.

L-LISTID

T-TERMID

A-APPLID

S-SESSION ID

Note: If you want to send the message to all TPX users, this field must be left blank.

NTPXALL (assembler)

NOTI-ALL-OPTION (COBOL)

Specifies whether you want to send the message to all TPX users. The possible values are:

N-(Default) Do not send the message to all TPX users.

Y-Send the message to all TPX users.

Note: If you select **Y**, the NTPXTYPE (assembler) or NOTI-USERID-OPTION (COBOL) field must be left blank.

NTPXSAVE (assembler)

NOTI-SAVE-OPTION (COBOL)

Specifies whether the message will be saved for the user. The possible values are:

Y-(Default)Save the message for the user.

N-Do not save the message for the user.

NTPXBREK (assembler)

NOTI-BREAK-IN-OPTION (COBOL)

Specifies whether you want users in an active TPX session to receive messages immediately. The possible values are:

N-(Default) The Notification utility will not interrupt active TPX sessions.

Y-The Notification utility will interrupt active TPX sessions and display the message. The user can then decide to save or delete the message before returning to the session.

XMIT-specific Parameters

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

NXMTNODE (assembler)

NOTI-NJE-NODE (COBOL)

The site-specific NJE target node name. This is a required entry

NOTERMSG (assembler).

NOTI-XMIT-MESSAGE (COBOL)

The Notification utility fills in this field if there has been an error. This field should be moved to the exit block error message field.

NXMTUNON (assembler only)

Specifies whether you want the Notification utility to append the NONOTIFY operand to the XMIT command. The possible values are:

N-(Default)Do not append the NONOTIFY operand to the XMIT command.

Y-Append the NONOTIFY operand to the XMIT command.

The Email Interface Program BC1PMLIF

You can write free format email messages to be sent during CA Endevor SCM processing. The email interface program BC1PMLIF allows you to do this without having an in depth knowledge about the $NOTIFY block and or the ESMTP table lookup.

Your program can call BC1PMLIF at exit points or in processors with an easy to use parameter list. BC1PMLIF formats the required internal control blocks and calls the existing CA Endevor SCM email interface modules. The following parameter list is passed to BC1PMLIF. The first two parameters are mandatory, and the others are optional.

MYSMTP-MESSAGE (132 bytes)

Error message return area

MYSMTP-USERID (8 bytes)

Mainframe USERID used to lookup the email address in ESMTPTBL

MYSMTP-FROM (50 bytes)

Email FROM line, no embedded spaces are allowed.

MYSMTP-SUBJECT (50 bytes)

Email SUBJECT line, embedded spaces are allowed.

MYSMTP-TEXT

Two byte counter followed by nn lines of data. Up to 99 lines of 133 bytes are allowed. Email line count, Email text area.

MYSMTP-URL (1 byte)

Includes the default URL value of ESMTPTBL in the email. If no default URL is coded in ESMTPTBL (DFTURL=,) then no URL is included in the email. Can contain Y or any other 1-byte value; the content of the passed value is not important.

BC1PMLIF uses the CA Endevor SCM ESMTPTBL to translate the Mainframe User ID name into an email address. Then it formats the message and sends it to the email server. If BC1PMLIF detects any errors during execution, it sets a non-zero return code and posts an error message in the Error Message area.

Note: A sample COBOL program that makes use of BC1PMLIF is provided as member CALLMLIF in the installation source library. For more information, see the *Exits Guide*.

Chapter 10: Using the Point in Time Recovery Feature

This section contains the following topics:

Point in Time Recovery

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

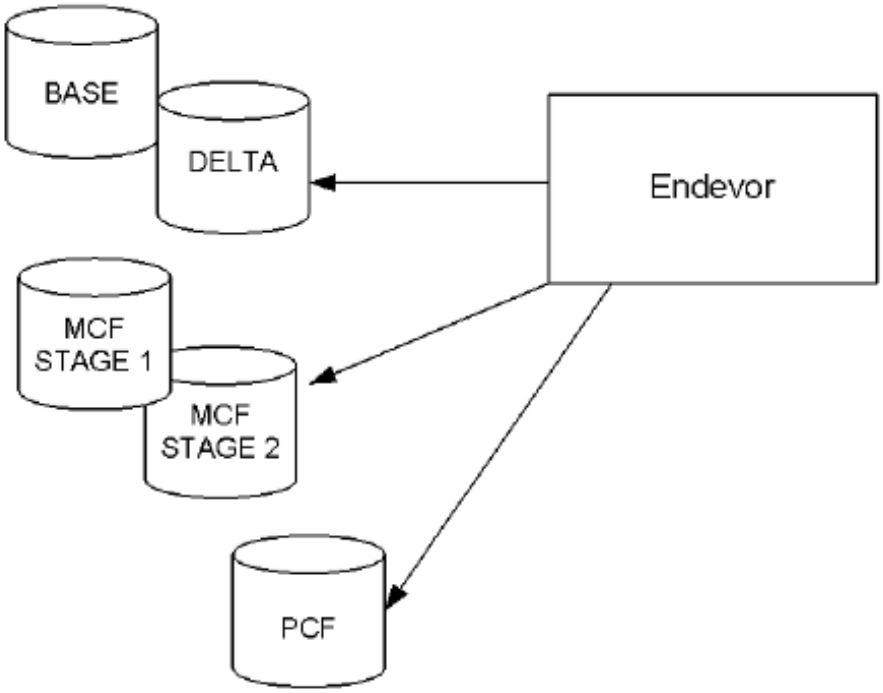
PITR allows you to recover changes made since the last backup to Element Catalog (ECF), Element Catalog Index (EIX), CA Endevor SCM Package Control File (PCF), Master Control File (MCF), and base and delta libraries.

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

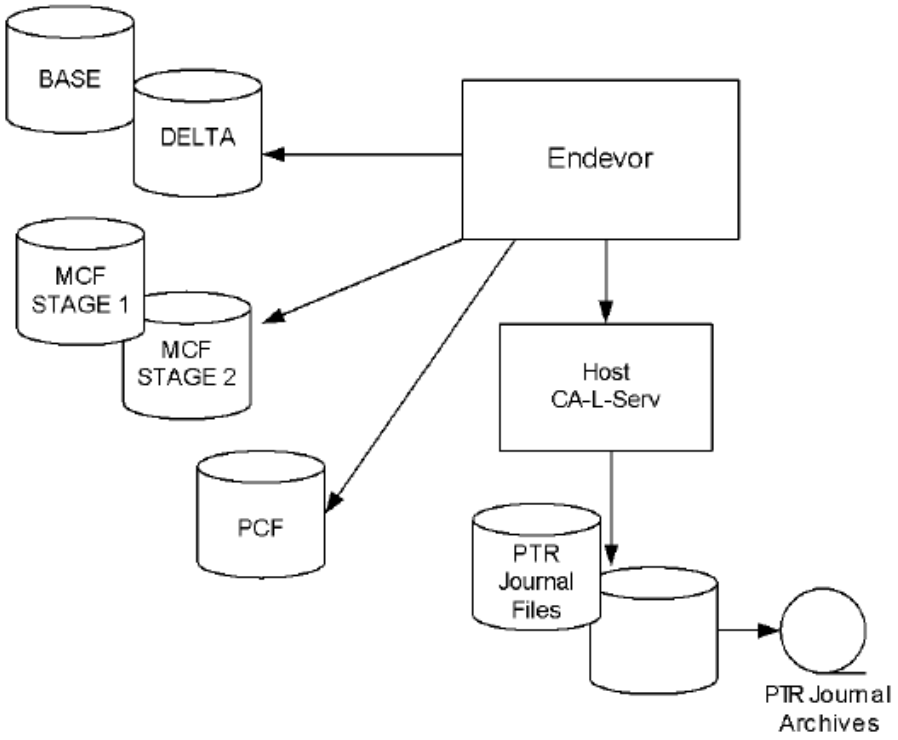
PITR builds GENERATE SCL for elements that have been affected by the recovery process.

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

The following illustrates CA Endevor SCM without PITR:



The following illustrates CA Endevor SCM with PITR:



CA L-Serv must be enabled in order to utilize the PITR feature. It is recommended that when implementing the PITR feature you have available someone who is familiar with how CA L-Serv is set up at your site. You should have access to the following guides as well:

- *CA Common Services for z/OS Getting Started Guide*.
- CA Common Services for z/OS *Administration Guide*.

How CA L-Serv Manages PITR Journal Files

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

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

Note: For more information about file groups, see the CA Common Services for z/OS *Administration Guide.*

How to Activate Journaling

After you define the journal data sets to CA L-Serv, you activate journaling for the following:

- Site level changes (the Package Control File element catalog and element catalog index) by specifying a CA L-Serv Group ID in the JRNLGRP parameter in the TYPE=MAIN section of the C1DEFLTS table.

  Note: The TYPE=MAIN JRNLGRP parameter is required if PITR is activated.

- Element and environment definition changes (Master Control File, base and delta libraries) by specifying a CA L-Serv Group ID in the JRNLGRP parameter in the appropriate TYPE=ENVIRONMENT section of the C1DEFLTS table.

  Note: For more information about how to specify group IDs in the C1DEFLTS table, see How to Modify the C1DEFLTS Table (see page 158).

Journaling

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

There are three steps in the journaling process:

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

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

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

Example: Using the Journaling Process

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

1. PROGX (1.0) copied back to Stage 1.
2. The MCF record, element base and element delta for PROGX (1.0) are written to the journal.

3. PROGX (1.0) compared with modified version being added, and a delta is created.

   The MCF record, element base and element delta for PROGX (1.1) are written to the journal.

4. The generate processor is executed.

   The PROGX (1.1) MCF record is written to the journal.

5. The component list is updated.

   The PROGX (1.1) MCF record, component base and component delta are written to the journal.

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

How to Offload Journal Data Sets

You can instruct CA L-Serv to automatically submit a batch job when the current journal data set in a group becomes full. This job offloads the information to an archive file, reinitializes the data set, and begins writing to the next data set in the group.

Note: For more information about how to set up CA L-Serv to submit this job automatically, see How to Define Journaling Components to CA L-Serv (see page 156).

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

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

The Recovery Utility

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

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

Note: The Recovery utility uses the recovery utility control statements to determine which transactions to recover. For more information, see Recovery Utility Syntax (see page 165).

The offloaded journal data sets can be concatenated in any order since all journal log entries are sorted prior to processing.

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

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

How to Enable Journaling

The steps required to enable journaling are listed next and described in detail in the following sections.

- Step 1. Determine Naming Conventions
- Step 2. Write Archive JCL
- Step 3. Allocate Journal and Archive Data Sets

- Step 4. Define the Journaling Components to CA L-Serv
- Step 5. Modify the C1DEFLTS Table

How to Determine Naming Conventions

The first step in enabling journaling is to decide on a naming convention for the following:

- PITR journal data sets.
- Journal group IDs used to reference groups of journal data sets.
- Generation data groups (GDGs) to be used as archive files.

Note: The naming conventions presented next are only suggestions.

Consider including the group ID in the names of both the journal and the archive files. For example, four journal files in a group named GRP1 might be named as follows:

```
NDVR.GRP1.JOURNAL1
NDVR.GRP1.JOURNAL2
NDVR.GRP1.JOURNAL3
NDVR.GRP1.JOURNAL4
```

The corresponding GDG might be named as follows:

```
NDVR.GRP1.ARCHDATA
```

CA L-Serv accesses the journal data sets using a journal group ID. Each journal group ID can reference one or more journal data sets. The data sets associated with a particular group ID can be the journal data sets for a Package Control File, the data sets for a particular environment, or a combination. Group names can be up to four characters in length.

The PITR process can be streamlined somewhat by using a single group ID for all the journal data sets at a site.

Note: For more information about file groups, see the *CA Common Services for z/OS Administrator Guide*. For suggestions on setting up groups for journal data sets, see Implementation Scenarios (see page 159).

How to Write Archive JCL

When CA L-Serv tries to write to a journal data set that is full, it marks the data set as unavailable and issues a console message.

When a data set becomes full, CA L-Serv automatically selects the next available journal data set for recording subsequent change activity. Optionally, you can instruct CA L-Serv to submit automatically a batch job to offload the journal information to a sequential file (for example, to a tape), switch to the next available journal data set, and reinitialize the journal data set that has just been offloaded.

The automatic offloading of journal data set information is implemented using the ARCHIVE command of the CA L-Serv LDMAMS utility.

The JCL for archiving full journal data sets is in member BC1JJARC in iprfx.iqual.CSIQJCL. To use this JCL, modify it as necessary and include it in the data set referenced in the JCLLIB DD statement of the CA L-Serv start-up member in your site PROCLIB.

Note: For more information, see the *CA Common Services for z/OS Administration Guide*.

How to Allocate Journal and Archive Data Sets

Journal files are VSAM ESDS data sets. Create these data sets with the utility IDCAMS.

Journal files should be allocated with a maximum record size of 32,760. Journal archive files should have an LRECL of 32,756, and a maximum length of 32,760.

The files to be used to archive full journal data sets are best set up as generation data groups (GDG).

Note: Separate journal data sets and journal archive GDGs should be defined for each CA Endevor SCM journal group ID.

Sizing Considerations

The size of the journal data sets depends on the amount of CA Endevor SCM activity at your site and the size of your elements. Here are some general guidelines to follow:

- Environment and package records take up a relatively small percentage of the total space.
- When a journal data set is full, it is archived. The next journal data set must be at least large enough to accept the journal records created during the time while the preceding data set is being archived.
- We recommend using three or four journal data sets. Remember that smaller journal data sets require less time to archive, making them available again for journaling more promptly.

For example, consider a site with an average element size of 1,000 lines, with 500 updates per day to these elements. Since most actions involve one write of an element base, element delta, component list base, and component list delta, and two writes of the MCF, a size calculation would appear as follows:

- Element base and delta, 1,000 lines @ 80 characters 80,000
- MCF update1, 100
- Component list base and delta, 100 lines @ 100 characters 10,000
- MCF update1, 100
- Total bytes, allowing for approximation 100,000

Assuming 45,000 bytes per track, this means that an average CA Endevor SCM action requires 2.5 tracks of journal data set space.

Assuming 500 transactions per day, there needs to be 1,250 tracks of journal data set space to handle one day's activity.

Allocating this space amongst three journal data sets means that each journal data set would be approximately 420 tracks. In this scenario, you might consider allocating a fourth journal data set to handle the package and environment definition updates.

Recommended Journal Data Sets

We recommend that at least two journal data sets, and preferably four, be allocated for each group. This allows CA L-Serv to switch to the second journal data set when the first data set becomes full and, optionally, to submit a batch job to offload and reinitialize the first journal data set.

The JCL for defining a generation data group and journal data sets is in member BC1JJDEF in the iprfx.iqual.CSIQJCL data set. Before submitting this job, perform the following actions:

- Specify the name for the GDG. Consider including the group ID of the related journal group in the GDG name.
- Change as necessary the maximum number of data sets to be retained in the GDG. The default in this example is 100.
- Change the names of the journal data sets in the job according to your naming standards.
- Specify values for cylinders (CYL) and volume (VOLSER).

Note: Do not specify a secondary extent for the cylinder allocation. Doing so prevents the journal files from filling up, defeating the purpose of journaling.

How to Define the Journaling Components to CA L-Serv

After allocating the journal files and defining a generation data group (GDG) for the archive files, you must define the journaling mechanism to CA L-Serv. Three members are involved in this process:

- The CA L-Serv PROC
- LDMPARM
- NDVRPARM

Note: For examples of how to set up journaling, see Implementation Scenarios (see page 159). For additional information, see the *CA Common Services for z/OS Getting Started Guide* and the *CA Common Services for z/OS Administration Guide*.

The CA L-Serv PROC

If you are using an existing CA L-Serv, you need to modify the PROC for that CA L-Serv. If you are creating a new L-Serv, you need to write a PROC for that CA L-Serv. This start-up procedure must include an //LDMCMND DD statement, which points to the CA L-Serv parameter data set.

Note: For details, see the *CA Common Services for z/OS Getting Started Guide*.

LDMPARM

LDMPARM is the default name of a member in the CA L-Serv parameter data set that contains the parameters for the CA L-Serv to which you want to identify the PITR components. If you are using an existing CA L-Serv, you need to modify the existing LDMPARM member for the CA L-Serv you want to use to include an ATTACH statement specifying the CA L-Serv type (local, remote, or host) and an INCLUDE statement referencing the member NDVRPARM. If you are using a new CA L-Serv you need to create this member.

An example of the syntax for the statements to be included in the LDMPARM member is shown next:

```
ATTACH
FILESERVER,SERVERTYPE=HOST,COMMSERVER(operands)
INCLUDE NDVRPARM
```

When included in a new or existing LDMPARM member, this syntax specifies a Host CA L-Serv, with L-Serv's communication server enabled, and specifies NDVRPARM as the member identifying the data sets to be managed by this CA L-Serv.

Note: For details, see the *CA Common Services for z/OS Getting Started Guide*.

NDVRPARM

NDVRPARM is the suggested name for a member in the CA L-Serv parameter data set that contains an ADDFILE command for each CA Endevor SCM library or PITR journal data set managed by L-SERV. A sample NDVRPARM member in an instance where CA L-Serv is being used to manage journal data sets is shown next:

```
************************************************************
***  JOURNAL FILES
************************************************************
ADDPOOL 13 (32768,100)
 ADDFILE JRNL1 uprfx.uqual.ugrpid.JOURNAL1 GROUP=ugrpid,
  OPTION=(SUBMIT) POOL=13,
  JCLMEMBER=BC1JJARC
 ADDFILE JRNL2 uprfx.uqual.ugrpid.JOURNAL2 GROUP=ugrpid,
  OPTION=(SUBMIT) POOL=13,
  JCLMEMBER=BC1JJARC
 ADDFILE JRNL3 uprfx.uqual.ugrpid.JOURNAL3 GROUP=ugrpid,
  OPTION=(SUBMIT) POOL=13,
  JCLMEMBER=BC1JJARC
 ADDFILE JRNL4 uprfx.uqual.ugrpid.JOURNAL4 GROUP=ugrpid,
  OPTION=(SUBMIT) POOL=13,
  JCLMEMBER=BC1JJARC
```

The ADDFILE and GROUP= clauses are required for journal files. The OPTION=(SUBMIT) and JCLMEMBER= clauses are required for automatic archiving. The ADDPOOL and POOL= clauses are optional but recommended.

Note: Do not use the APPEND parameter of the ADDFILE statement when identifying journal files or CA Endevor SCM files to CA L-Serv.

Note: For complete syntax for the ADDFILE and ADDPOOL commands, see the *CA Common Services for z/OS Administration Guide*. For information about using CA L-Serv to manage CA Endevor SCM libraries, see the *CA Common Services CA L-Serv Technical Bulletin*.

How to Modify the C1DEFLTS Table

In addition to identifying the journaling components to CA L-Serv, you also must modify the C1DEFLTS table to include the journal group IDs for the journal files, and to identify the subsystem name associated with the CA L-Serv address space being used to implement journaling.

The syntax for including this information in the C1DEFLTS table is shown:

JRNLGRP=(*group id,nnnn*)

The following describes the syntax:

group id

The journal group ID associated with the journal files.

nnnn

The subsystem name for CA L-Serv being used to implement journaling. The default CA L-Serv subsystem ID is LSRV.

You must include JRNLGRP= statements in the following sections:

1. The TYPE=MAIN section of the C1DEFLTS table to enable journaling in the Package Control File.

   Note: The TYPE=MAIN JRNLGRP parameter is required if PITR is activated.

2. Each TYPE=ENVIRONMENT section of the MCF base and delta libraries (VSAM as well as non-VSAM) for which you want to enable journaling.

Note: If you want to use the same journal group ID for more than one environment, you must include the journal group ID in each environment section of the C1DEFLTS table. We recommend that PITR be specified for all environments defined in the C1DEFLTS.

Journal Group Examples

Examples of the JRNLGRP= parameter syntax to be included in the TYPE=MAIN section and the TYPE=ENVIRONMENT sections of the C1DEFLTS are shown next:

```
C1DEFLTS TYPE=MAIN,           X
  JRNLGRP=(group id,nnnn) GRP ID/SUBSYS NAME FOR PITR X

C1DEFLTS TYPE=ENVRNMNT,          X
  JRNLGRP=(group id,nnnn) GRP ID/SUBSYS NAME FOR PITR X
```

Reassemble the C1DEFLTS Table

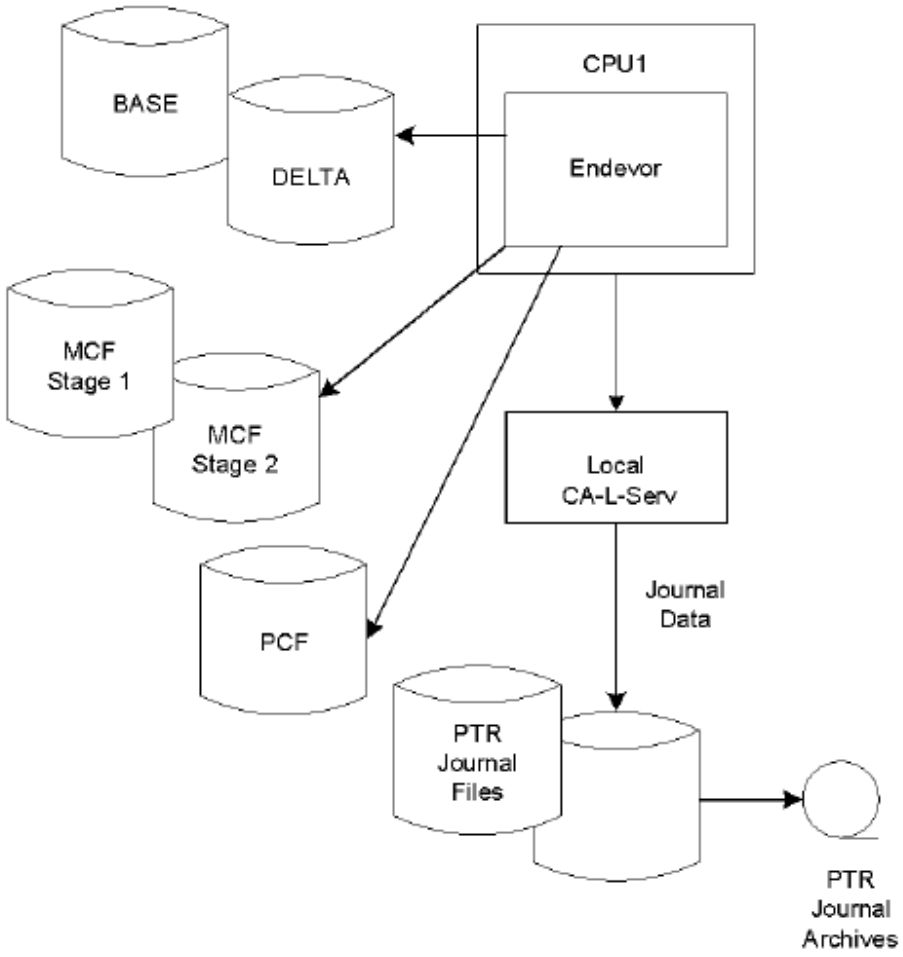
After including the group IDs and CA L-Serv subsystem names for the journal data sets, you must reassemble the C1DEFLTS table in order to enable journaling.

Implementation Scenarios

The number of CA L-Servs required to implement PITR depends on the site configuration. This section presents common single and multiple CPU implementation scenarios, and provides guidelines for each.

Single CPU Implementation

The following is an example of a single CPU implementation. This configuration is suitable for sites where CA Endevor SCM is running on a single CPU.



How to Implement a Single CPU

The steps to implement this configuration are:

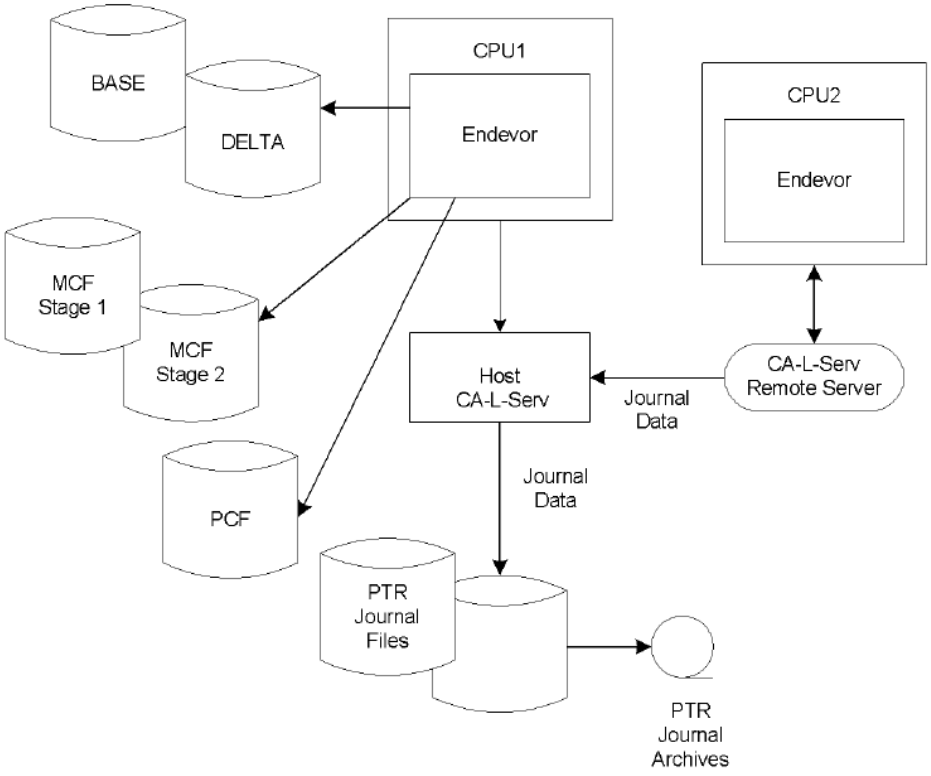
1. Define CA L-Serv as Local, using the default subsystem name LSRV.
2. Create or modify member NDVRPARM, LDMPARM, and the CA L-Serv PROC.

   Note: Follow the instructions earlier in this section, and refer to the *CA Common Services for z/OS Getting Started Guide* as necessary.
3. Modify the C1DEFLTS table.

   Note: For more information about modifying the C1DEFLTS table, see How to Modify the C1DEFLTS Table (see page 158).

Multiple CPU Implementation Using Remote Journaling

The following is an example of a multiple CPU implementation with all journaling taking place on CPU 1. This is referred to as remote journaling. Use this configuration at sites where there is relatively heavy CA Endevor SCM usage on the CPU where CA L-Serv is managing the journal files, and relatively light usage on the remote CPU (CPU 2).



How to Implement a Multiple CPU

The steps to implement this configuration are:

1. Set up CA L-Serv Host by:
   - Defining it as Host, using the subsystem name LSRV.
   - Creating or modifying members NDVRPARM, LDMPARM, and the CA L-Serv PROC, as described earlier in this section.
2. Set up CA L-Serv Remote by:
   - Defining it as Remote, using the subsystem name LSRV.
   - Creating or modifying member LDMPARM, and the CA L-Serv PROC, as described earlier in this section. As a remote CA L-Serv, it does not manage any files and therefore does not need a NDVRPARM member.

3. Set up communication between CA L-Serv Host and CA L-Serv Remote, referring to the *CA Common Services for z/OS Getting Started Guide* as necessary.
4. Modify the C1DEFLTS table.

   Note: For more information about modifying the C1DEFLTS table, see How to Modify the C1DEFLTS Table (see page 158).

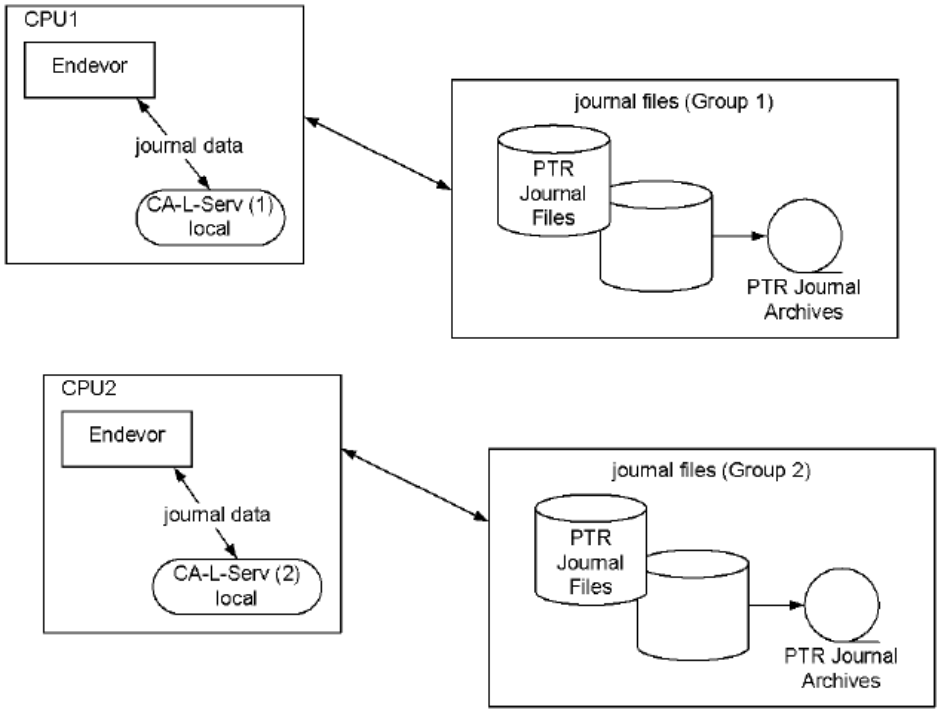
Performance Considerations for a Multiple CPU Implementation Using Remote Journaling

Because it requires only one set of journal data sets, this scenario represents the easiest way to set up PITR in a multiple CPU operating environment.

There is a trade-off to this scenario, namely that handling journaling remotely is somewhat slower than local journaling.

Multiple CPU Implementation Using Local Journaling

A Multiple CPU implementation with CA L-Serv controlling both the journal files and the ELIBs, with journaling taking place on each CPU is shown next. Use this configuration at sites where there is relatively heavy CA Endevor SCM usage on both CPU 1 and CPU 2.



How to Implement Multiple CPU Using Local Journaling

The steps to implement this configuration are:

1. Set up CA L-Serv 1 and CA L-Serv 2 by:
   - Defining both as Local, using the same subsystem name for each.
   - Creating or modifying members NDVRPARM, LDMPARM, and the CA L-Serv PROC, as described earlier in this section.
2. Modify the C1DEFLTS Table to include entries for CA L-Serv 1 and CA L-Serv 2.

   Note: For more information about modifying the C1DEFLTS table, see How to Modify the C1DEFLTS Table (see page 158).

Performance Considerations for a Multiple CPU Implementation Using Local Journaling

Because this scenario requires two groups of journal data sets, performing periodic backups is somewhat more complicated. The trade-off is that journaling executes somewhat more quickly when it is performed locally.

Perform Periodic Backups of CA Endevor SCM

When journaling is enabled, it is important to manage the archive files carefully. Perform periodic backups before the GDG becomes full.

To perform periodic backups of CA Endevor SCM

1. Clean out journal data sets that may contain information, by executing the JCL found in member BC1JJARG in the data set iprfx.iqual.CSIQJCL.
2. Disable journaling by either issuing the CA L-SERV REMOVEFILE command or stop CA L-Serv using one of the following console commands:

   ```
   /F task,SHUTDOWN
   /P task
   ```

   These commands deactivate all functions running in the CA L-Serv address space. For information on CA L-Serv system commands refer to the *CA Common Services for z/OS Getting Started Guide*.
3. After you have successfully removed the files from the control of CA L-SERV, you must change the SHR(1 3) to SHR(3 3). To do this, run job BC1JLSRV supplied in your iprfx.iqual.CSIQJCL library. This job is intended to ALTER the SHR attributes of the Master Control File (MCF), Package and ELib datasets from SHR (1 3) to SHR (3 3).
4. Backup CA Endevor SCM, using either the Unload Utility or your regular backup utility.

5. Delete all the PITR Journal Archives.
6. Restart CA L-Serv by either using the following following START command, or by using the CA L-SERV ADDFILE command to issue journaling again.

   ```
   /S task parms
   ```

Perform Point in Time Recovery

If a CA Endevor SCM Package Control File, Master Control File, base library or delta library is lost, you can perform a point in time recovery. Before performing Point in Time Recovery, make sure that there are no empty GDGs.

To perform a point in time recovery

1. Execute the CA L-Serv LDMAMS utility to offload all used journal data sets.
2. Disable PITR journaling to keep any new journaling from being issued.
3. Restore all CA Endevor SCM data sets to be recovered from the most current back-up (either CA Endevor SCM UNLOAD or similar back-ups).
4. Execute the CA Endevor SCM Recovery utility.

The following sections discuss these steps in detail.

How to Execute the CA L-Serv LDMAMS Utility

The next step is to offload all used journal data sets to sequential data sets. The sample JCL for offloading journal data sets can be found in member BC1JJARG in iprfx.iqual.CSIQJCL. Before using this sample JCL, do the following:

- Replace UGRPID with the CA L-Serv group ID for the journal data sets.
- Replace uprfx.uqual.ugrpid.JRNLDATA with the GDG name used at your site for the archive data sets.
- If the CA L-Serv load library has not been included in LINKLST, specify the load library data set name in the STEPLIB DD statement. Otherwise remove the STEPLIB DD statement.
- Specify the unit, volser, and space parameters for the journal archive data set.

The ARCHIVE statement in this sample differs from the statement used to archive data sets that are full. The difference is the SWITCH parameter. When a data set in a CA L-Serv group is not full, this parameter tells CA L-Serv to switch to the next data set in the group after archiving whatever data is in the data set.

Note: For more information, see the CA Common Services for z/OS *Administration Guide*.

How to Disable Point in Time Recovery Journaling

Disable journaling by either issuing the CA L-SERV REMOVEFILE command or by using one of the following console commands:

```
/F task,SHUTDOWN
/P task
```

Note: These commands deactivate all functions running in the CA L-Serv address space. For more information about CA L-Serv system commands, see the *CA Common Services for z/OS Getting Started Guide*.

How to Restore the Data Sets to Be Recovered

The first step in the Point in Time Recovery process is to restore the data sets that have been lost to the point of their most recent backup. This makes them available to the Recovery utility.

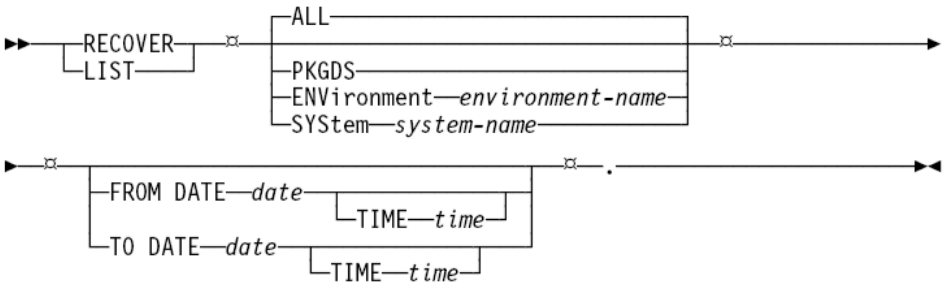
The utility uses the restored data sets as a baseline, recovering any transactions that have taken place since the most recent backup. Do this using the recovery procedure at your site.

How to Execute the Recovery Utility

The Recovery utility (program BC1PJRCV) uses the offloaded journal information to recover transactions that have occurred since the last backup.

Recovery Utility Syntax

The following is the syntax for the recovery utility:



This syntax is described as follows:

RECOVER /LIST

(Required) The RECOVER or LIST keyword must be the first word in the syntax. Use LIST if you want to list the contents of a journal file before recovering it.

PKGDS

(Optional) Indicates that you want to recover the package data set for the site.

ENVIRONMENT

(Optional) Indicates that you want to recover the named environment.

SYSTEM

(Optional) Indicates that you want to recover the named system.

ALL

(Default) Indicates that you want to recover the element catalog, element catalog index, package data set, MCF, base and delta library information.

FROM DATE, TIME

(Optional) Allows you to specify a date and time from which you want to start recovering information.

TO DATE, TIME

(Optional) Allows you to specify a date and time through which you want to recover information.

Recover Examples

To recover the package data set for a site, submit the following syntax:

```
RECOVER PKGDS.
```

To recover MCF, base and delta information for environment PROD, submit the following syntax:

```
RECOVER ENV PROD.
```

To recover MCF, base and delta information for environment PROD, system Finance, submit the following syntax:

```
RECOVER ENV PROD SYS FINANCE.
```

If, for some reason, the PROD environment should be restored as it was on July 19th (as opposed to all logged changes) the following clause would be specified:

```
RECOVER ENV PROD
TO DATE 19JUL92 TO TIME 08:00.
```

The following statement would recover all logged changes (contained in the input sequential journal files) for the Package Control File and for all environments:
`RECOVER ALL.`

You can find the JCL to run the recovery utility in member BC1JJRCV in iprfx.iqual.CSIQJCL.

The statements in the JCL are described in the following:

C1MSGS1

The C1MSGS1 DD statement specifies the destination for the Journal Recovery Execution Report.

C1SUMMRY

The C1SUMMRY DD statement specifies the destination for the Journal Recovery Execution Summary Report.

JOURNAL

The JOURNAL DD statement identifies the data set containing the journal data to be recovered.

C1SCL1

The C1SCL1 DD statement identifies the data set containing the SCL statements written by the Recovery utility.

BSTIPT01

The BSTIPT01 DD statement contains syntax for this run of the Recovery utility.

The Journal Recovery Execution Report

The Recovery utility automatically produces a two-part Journal Recovery Execution report. The Journal Recovery Execution Log contains the following sections:

- Transaction detail
- Transaction summary

The Journal Recovery Execution Summary contains the following sections:

- Data set activity summary
- SCL statement summary
- Processor execution summary

The Journal Recovery Transaction Detail Report

The transaction detail section of the Journal Recovery Execution Log reports on each journal transaction that is recovered. The following shows the messages contained in the transaction detail section and the information provided by each message:

JRCV032I

The transaction number, the journal date and time stamps, and the number of records in the transaction.

JRCV015I

The kind of action.

JRCV016I

The CA Endevor SCM location associated with the member in the previous message.

JRCV031I

The date and time of the most recent update to the CA Endevor SCM entity described in this entry. If no information is available, a NOT FOUND message is returned.

JRCV023I

Indicates that generate SCL has been built for the entity.

JRCV025I

Indicates that the recover utility has issued a generate processor request for the element.

JRCV026I

The CA Endevor SCM location associated with the element to be generated.

JRCV020I

Indicates that the journal entry has been recovered, and provides the return code for the recovery process.

The Journal Recovery Journal Input Record Summary

The Journal Input Record Summary lists the number of records of each record type handled by the recovery utility. The following are the abbreviations used in this report:

PCF

Package Control File.

MCF

Master Control File.

EBASE

Element base.

EDELTA

Element delta.

CBASE

Component list base.

CDELTA

Component list delta.

The Journal Recovery Data Set Activity Summary

The data set section of the Journal Recovery Execution Report describes the activity recorded in the journal file for each data set in the journal file. The messages contained in this section are described in the following:

JRCV100I

The data set usage by CA Endevor SCM. Usage may be any of the following:

ECF

Element Catalog

EIX

Element Catalog Index

CBASE

Component list base

CDELTA

Component list delta

EBASE

Element base

EDELTA

Element delta

MCF

Master Control File

PCF

Package Control File

JRCV113I

The CA Endevor SCM location associated with the file. For example, a file used to store element base records might be used for base records associated with a particular environment, stage, system, and type. A Master Control file on the other hand is associated with only an environment and stage.

JRCV101I

Indicates the number of members created in this data set for the period covered by the journal file.

JRCV102I

Indicates the number of members updated in this data set for the period covered by the journal file.

JRCV103I

Indicates the number of members deleted in this data set for the period covered by the journal file.

The Journal Recovery SCL Statement Summary

The SCL statement summary portion of the Journal Recovery Execution Report shows the number of GENERATE statements written by CA Endevor SCM location (environment, stage) and inventory classification (system, subsystem, and type).

Chapter 11: Using the Search and Replace Utility

This section contains the following topics:

Search and Replace Utility

The Search and Replace utility lets you search for and optionally replace the following:

- text strings in elements (see page 171)
- output data set names in component lists (see page 208)

You can run the utility in validate mode to check that the SCL statements generated for your search request are valid. Whether the utility is run in replace or validate mode, you can generate a report of the search targets found. Additionally, other options can be specified.

Searching and Replacing Text Strings

The option to update text within elements enables you to search for a character string in elements that are under CA Endevor SCM control. You have the option to replace the character string with a different character string. If you specify a replacement string, the element is updated and added back into the entry stage at the environment specified on the request.

The Search and Replace utility is controlled by the SEARCH ELEMENT request (available in batch only). You specify the inventory location to be searched as well as any additional selection criteria, such as CCID or processor group. You also provide the character string for which you are looking and, optionally, a character string with which you want to replace the original string.

If you include a replacement string in your request, the utility replaces the original character string with the second string. The utility then adds the element back into the entry stage at the environment specified on the request. If you do not specify a replacement string, the element is not updated.

The premise of the Search and Replace Utility is that one or more text strings can be found and, optionally, replaced by one or more different text strings. Information is compared on a line-by-line basis, within specific columns. Information is replaced within specific columns of specific lines.

The following concepts are also important to your understanding of the search and replace process.

- IN-COLUMNS and BOUNDS ARE parameters are used in conjunction with each other to set the limits for the search and replace operations.

  Note: For a discussion of compare column ranges, IN COLUMNS rules, and BOUNDS ARE rules, see Text Replacement (see page 191).

- Data is manipulated only within the modifiable range. Data in columns outside this range is neither modified nor affected by data shifting.

  Note: For a discussion and examples regarding replacement string length, see Shorter Replacement String (see page 193).

- You can override the type compare columns by assigning IN-COLUMNS and BOUNDS ARE values in the SEARCH ELEMENT statement. You cannot assign a value that exceeds the type definition compare column value. For example, you are using element type COBOL, whose compare columns are 7-72. If you assign a right boundary of 80, you will receive an error message. The right boundary can be any value up to and including 72. Similarly, assigning a left boundary of 6 results in an error. The left boundary value cannot be less than 7.

  Note: For a discussion of the SEARCH ELEMENT SCL, see Search Element SCL (see page 182). For a discussion about IN COLUMNS and BOUNDS ARE rules, see Text Replacement (see page 191).

How the Search and Replace Utility Begins the Search

The Search and Replace utility begins its search at the inventory location indicated in the SEARCH ELEMENT request. If the element is not found at that location and if you requested that all environments in the map be searched, the utility checks subsequent environments for the element. Once found, the utility processes only the first occurrence of the element. Remaining stages are neither searched nor processed.

The search is always done against the current level of the element, and always begins at the entry stage of the indicated environment. You cannot specify a beginning stage location in your SEARCH ELEMENT request.

The Search String

The search string can be from 1-72 characters in length. You cannot use a string that contains both single quotation marks and double quotation marks.

By default, the Search and Replace utility looks for the character string in the column range associated with the element type entered in the request. You can override the type values by coding an explicit search column range.

Search and Replace Utility Processing Modes

The Search and Replace utility executes in one of three modes:

Validate

Executed when you code VALIDATE in the PARM= field in the execution JCL. The utility parses and verifies the SCL statements in the SEARCH ELEMENT request.

Note: For more information about validate mode, see Validate Mode (see page 176).

Search-Only

Executed by default, the utility searches for the search string specified and produces a report indicating the element in which the string is found. Optionally (depending on what is coded in the execution JCL), the utility generates SCL for each element that contains the search string.

Note: For more information about search-only mode, see Search-Only Mode (see page 177).

Replacement

Executed when a replacement text string is included in the request and the OPTIONS UPDATE ELEMENT clause is coded. Both conditions must exist in order for the utility to replace one string with another. The utility replaces the original string with the replacement string, and adds the element back into the entry stage of the environment specified on the SEARCH ELEMENT request.

Note: For more information about replacement mode, see Replacement Mode (see page 179).

Search and Replace Column Definitions

There are several terms used to describe the search columns and replace columns. It is important that you understand each term as you work with the Search and Replace Utility. These terms are described as follows:

Compare columns (or type compare columns)

The compare columns associated with the element type definition. Each type (for example, COBOL, Assembler, or JCL) has specific columns within which CA Endevor SCM looks to identify changes. For example, the compare column range for COBOL is 7-72.

BOUNDS ARE parameters (or bounds or boundaries)

The columns within which CA Endevor SCM can place the new text string. BOUNDS ARE parameters begin with *left-column* and end with *right-column*. If omitted, the type definition compare columns are used.

The BOUNDS ARE parameters usually define the modifiable range.

IN COLUMNS parameters (or search columns)

The columns within which CA Endevor SCM searches for a particular text string. IN COLUMNS parameters begin with *from-column* and end with *to-column*. If omitted, the BOUNDS ARE values are used.

Modifiable range

The union of the BOUNDS ARE parameters and IN COLUMNS parameters.

- The left or first position of the modifiable range is the IN COLUMNS from-column.
- The right or end position of the modifiable range is BOUNDS ARE right-column, with one exception:

  If the IN COLUMNS to-column is greater than the BOUNDS ARE right-column, the IN COLUMNS to-column value is used as the end of the modifiable range.

The default modifiable range is the compare column range of the element type.

Search and Replace Utility Operating Considerations

This section details operating considerations that pertain to the Search and Replace utility.

Miscellaneous Operating Considerations

Miscellaneous operating considerations include the following:

- All updates to elements are performed at the entry stage in the environment specified in the SEARCH ELEMENT request.
- Variable length records are never shortened. A record's length may increase, but it will not decrease.
- In replacement mode, the Search and Replace utility does not search elements that are signed out to users other than the current user, unless the OVERRIDE SIGNOUT option is specified in the SEARCH ELEMENT request. Nor does the utility search elements in an in-between stage (that is, a stage not on the map but between two stages that are on the map).

The Search and Replace utility uses the CA Endevor SCM security system to verify that a user is authorized to perform the requested actions against the element. The following security checks are performed:

- Does the user have RETRIEVE authority for the element at the inventory location at which it is found?

  This check is performed before searching an element.

- Does the user have OVERRIDE SIGNOUT authority at the inventory location at which the element is found?

  This check is performed when the OVERRIDE SIGNOUT clause is specified and the element is not signed out to the current user. (Applies to replacement mode only.)

- Does the user have ADD or UPDATE authority for the element at the entry stage of the specified environment?

  This check occurs when the updated element is added back into CA Endevor SCM. If the element exists at the entry stage, the user must have UPDATE authority for the element at that stage. If the element does not exist at the entry stage, the user must have ADD authority for the element at that stage.

Serializing the Element

The Search and Replace utility puts a “lock” on an element when it is being processed. The lock is placed at the environment indicated in the SEARCH ELEMENT request and at the source environment, if the element was found up the map. Therefore, other CA Endevor SCM actions against the element, such as SIGNOUT or RETRIEVE, may be prohibited while you are processing the element.

Serializing the element applies to replacement mode only.

Exits Invoked by the Search and Replace Utility

The Search and Replace utility invokes three CA Endevor SCM exits:

- Exit 1--Security
- Exit 2--Before Action
- Exit 3--After Action

Exits are handled as follows:

Exit 1

Invoked for the following actions:

- Retrieve
- Add, Update

When invoked:

- Before any element processing begins.
- Before ADD or UPDATE processing, only if it has been determined that the element will need to be added back into CA Endevor SCM.

Exit 2

Invoked for the following actions: Add or Update

When invoked: Before action processing begins.

Exit 3

Invoked for the following actions: Add or Update, based on exit 2

When invoked: After a corresponding (and successful) invocation of exit 2.

- If exit 2 is invoked and allows an action, exit 3 is invoked after the action has been performed.
- If exit 2 is invoked but does not allow an action, exit 3 is not invoked.
- If exit 2 is not invoked for an action or element, exit 3 is not invoked for that action or element.

Validate Mode

The Search and Replace utility operates in validate mode when you code VALIDATE as part of the PARM= parameter in the execution JCL. In this mode, the utility parses and verifies the generated SCL statements for proper syntax. If no errors are found, the SCL statements are formatted.

Processing stops after validation. The actions implied by the SCL statements are not performed. Errors other than syntax errors are not noted at this time.

To invoke validate mode, you need to code the following in the execution JCL:

```
PARM='ENBS1000VALIDATE'
```

You can abbreviate the word validate, using any of the following entries:

```
V, VA, VAL, VALI, VALID, VALIDA, or VALIDAT
```

Search-Only Mode

The Search and Replace utility executes in search-only mode by default. In this mode, the utility searches for a particular search string in the elements specified in your SEARCH ELEMENT request. At a minimum, the utility generates a list of the elements that contain the search string. Depending on what you code in the request or in the execution JCL, the utility performs the following actions:

OPTIONS LIST DETAILS in the request

Displays the line of the element that contains the search strings.

A replacement string and OPTIONS LIST DETAILS in the request

Displays the line of the element that contains the search string followed by the same line containing the replacement string.

A replacement string in the request and allocate the ENSSCLOT file in the JCL

Generates SCL for all the elements that would be affected by replacement of the search string. SCL statements are generated only when a replacement string is specified in the original search request.

Note that in search-only mode, the OPTIONS UPDATE clause is not specified.

Search-Only Mode Processing

The Search and Replace utility determines the elements to be searched based on inventory location and additional selection criteria provided in the SEARCH ELEMENT request. For each element identified, the utility reads--on a record-by-record basis--the current level of the element, and searches for the search string. The utility produces a series of reports listing each element that was searched and the result of the search (for example, number of search string matches found in the element).

The Search and Replace utility verifies RETRIEVE authority before processing the element.

The Search and Replace utility does not do the following when in search-only mode:

- The utility does not update the element.
- The utility does not perform signout processing.
- The utility does not check whether an element is at an in-between stage (a stage not on the map, but between two stages that are on the map).

Important! The search string is case-sensitive. If the search string contains only uppercase characters, the utility looks only for text in uppercase characters. If a line in an element matches the text of the search string but is in lowercase or a combination of lowercase and uppercase characters, the utility does not record a match.

Generating Search Elements SCL

The Search and Replace utility provides the option of generating SCL statements for each element that contains the search string. These SCL statements can then be executed to perform actual replacement of the search strings.

This capability is useful when a large number of elements are searched before search strings are replaced. You can review the output to see how elements will be affected by the text replacement. If the output is acceptable, execute the Search and Replace utility again, using the generated SCL as the input file. Because the SCL statements have been created, the utility only needs to execute them; a second search of the entire inventory is not necessary. Using this option not only allows you to review the results of replacing text before you actually do so--it also saves you time.

The Search and Replace utility generates SCL statements if you enter a replacement string in the SEARCH ELEMENT request and you allocate the ENSSCLOT DD statement in the execution JCL. The SCL statements written to the ENSSCLOT file contain all the information entered in the original SEARCH ELEMENT request as well as an OPTIONS UPDATE ELEMENT clause. When you invoke the utility a second time, using the generated SCL as the input file, only the specified elements are searched and updated.

If the source SCL contains multiple SEARCH ELEMENT requests, the output data set may contain multiple SCL statements for the same element.

Note: Each FROM statement in the generated SCL will contain explicit system, subsystem, and type values, even if you used a wildcard for that value in the original SEARCH ELEMENT request. These values represent the location where the updates will be applied.

The ENSSCLOT File

The ENSSCLOT DD statement should refer to a sequential data set or a partitioned data set with an explicit member. Allocate the data set with the following attributes:

- DSORG=PS (or PO if a member name is specified)
- RECFM=F or FB
- LRECL=80

Replacement Mode

The Search and Replace utility operates in replacement mode when you include both the REPLACE WITH clause and the OPTIONS UPDATE ELEMENT clause in your SEARCH ELEMENT request. In this mode, the utility searches the element for the search string, replaces each occurrence of the string with the replacement string, and adds the element back into CA Endevor SCM at the entry stage of the specified environment.

Replacement Mode Processing

The Search and Replace utility determines the elements to be searched based on inventory location and additional selection criteria provided in the SEARCH ELEMENT request. For each element identified, the utility reads--on a record-by-record basis--the current level of the element, and searches for the search string. If the search string is found, the utility replaces that string with the replacement string specified in the SEARCH ELEMENT request. When the entire element has been searched and all relevant search strings replaced, the utility adds or updates the element to the entry stage of the environment indicated in your request.

In this mode, the utility verifies RETRIEVE, SIGNOUT, and ADD/UPDATE authorization. In addition, the utility checks whether any of the elements specified exist at an in-between stage (stage not on the map but between two stages that are on the map).

Note: For more information about these authorizations and conditions, see Processing Checkpoints (see page 180).

Be aware of the following:

- SIGNOUT authorization and in-between stage checking are not performed in search-only mode. Consequently, elements that are signed out to another user or exist at an in-between stage may be searched, and may have SCL statements written to the ENSSCLOT file. Because the utility does perform these two checks in replacement mode, some of the SCL statements may terminate with an error if you use this ENSSCLOT file as input to a subsequent Search and Replace job.
- The search string and the replacement string are case-sensitive. If the search string contains only uppercase characters, the utility looks only for text in uppercase characters. If a line in an element matches the text of the search string but is in lowercase or a combination of lowercase and uppercase characters, the utility does not record a match.

  Similarly, the utility places the replacement string in the element exactly as it has been coded in the SEARCH ELEMENT request. The utility does not convert lowercase characters to uppercase characters (or vice versa).
- When fetch processing occurs, if the value of Signout Fetch (SOFETCH), a CA Endevor SCM Defaults Table parameter, is Y, the element that is fetched will be signed out to you at the location from which it was fetched, unless it is already signed out to someone else. If the value of SOFETCH is N, the element that is fetched will not be signed out.

  The element that is put in the entry stage will be signed out to you.

Processing Checkpoints

Before and during processing, the utility checks for the following authorizations and conditions:

RETRIEVE authority

Before processing an element, if the user does not have RETRIEVE authority, then processing stops for that element and begins for the next element.

RETRIEVE authority

Before processing an element, if the user has RETRIEVE authority, then processing for the element continues.

SIGNOUT authority

After RETRIEVE authority is determined but before processing for the element, if the user does not have SIGNOUT authority, then processing stops for that element and begins for the next element.

SIGNOUT authority

After RETRIEVE authority is determined but before processing for the element, if the user does have SIGNOUT authority, then processing for the element continues.

Whether the element exists at an in-between stage

After RETRIEVE and SIGNOUT authority are determined but before the element is searched, if the element exists at an in-between stage, then processing stops for that element and begins for the next element.

Whether the element exists at an in-between stage

After RETRIEVE and SIGNOUT authority are determined but before the element is searched, if the element does not exist at an in-between stage, then processing for the element continues.

ADD/UPDATE authority

After the element is searched and the search string is replaced, but before the element is added or updated into CA Endevor SCM, if the user does not have ADD/UPDATE authority, then processing stops for that element and begins for the next element.

ADD/UPDATE authority

After the element is searched and the search string is replaced, but before the element is added or updated into CA Endevor SCM, if the user does have ADD/UPDATE authority, then the utility performs the appropriate action at the entry stage of the specified environment:

- UPDATE if the element exists at the entry stage
- ADD if the element does not exist the entry stage

Search and Replace Execution JCL

The Search and Replace control statements (SCL) are coded in the execution JCL used to activate the utility. The JCL which executes this utility can be found in member ENBSRPL1, in the JCL library iprfx.iqual.CSIQJCL.

The ENSSCLIN DD Statement

The user control statements specified in the ENSSCLIN DD statement are SEARCH ELEMENT requests, which specify element search criteria, the search string, and, optionally, a replacement text string. You can code as many SEARCH ELEMENT requests as you need; there is no defined limit on the number of statements allowed.

Note: For more information about the SEARCH ELEMENT SCL, see Search Elements SCL (see page 182).

The PARM= Statement

You must specify the ENBS1000 parameter to invoke the Search and Replace utility. You can optionally code the VALIDATE parameter (immediately after ENBS1000) to request validate mode processing.

The utility performs the following actions depending on the value that you specify in the PARM= statement:

PARM=ENBS1000

Parses and validates all the requests before processing them.

If the parser or validation routine detects an error, the utility will not execute any of the statements. The parsing routine attempts to parse all of the control statements before terminating.

PARM=ENBS1000VALIDATE

Parses and validates all the requests, but does not execute the requests even if no syntax or validation errors are found.This parameter must follow the ENBS1000 parameter, as shown here:

PARM=ENBS1000VALIDATE

Search Elements SCL

You can enter as many SEARCH ELEMENT requests as necessary in the control statement data set. There is no defined limit to the number of actions allowed in a single execution of the program.

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

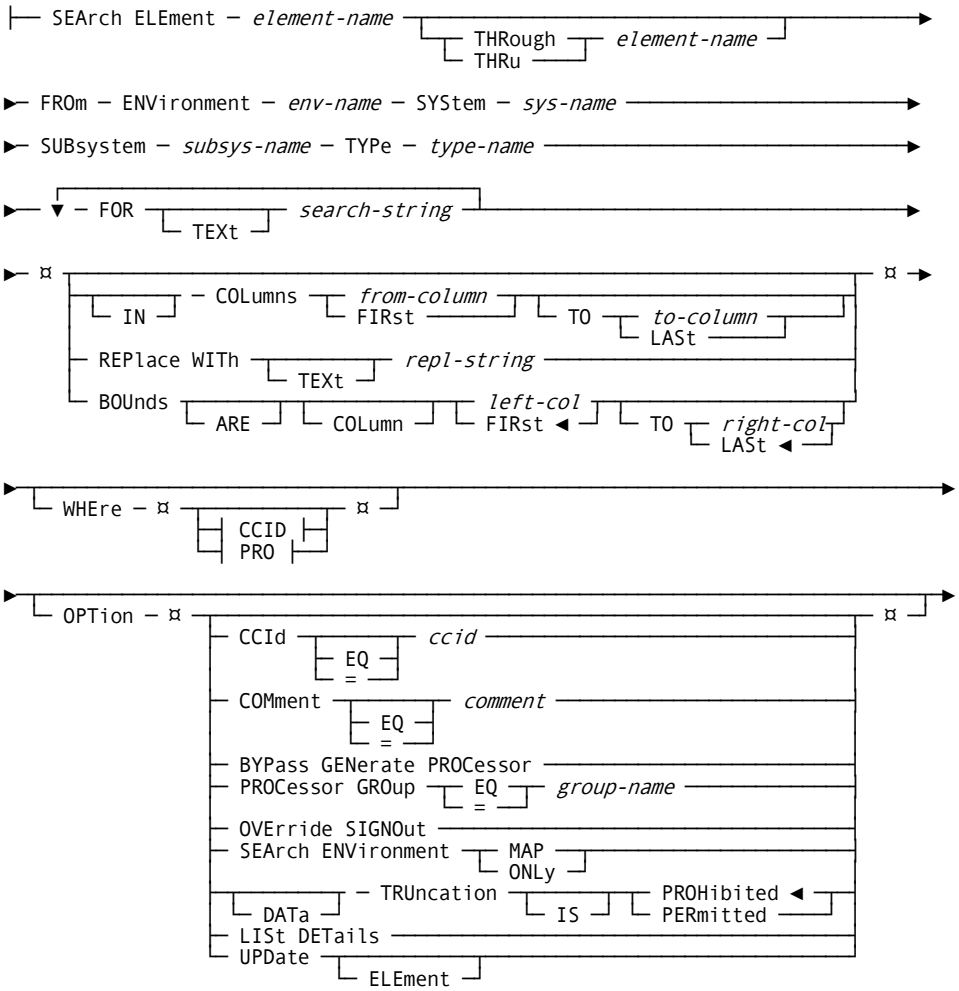
When the requests have been successfully parsed, the utility executes them. Requests are executed as long as the highest return code is less than or equal to 12.

Note: If you code PARM=VALIDATE, the utility will not execute the requests when parsing is complete.

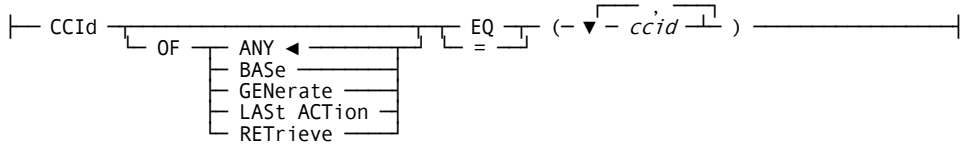
The SEARCH ELEMENT syntax is shown in the next section, followed by a description of each clause in the syntax.

Search Elements Syntax

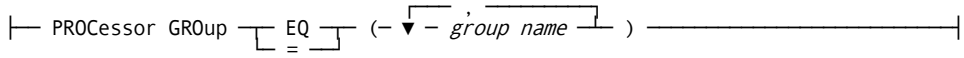
The following is the SEARCH ELEMENT syntax:



Expansion of CCID



Expansion of PRO



The Search Element Clause

The SEARCH ELEMENT clause allows you to specify one or more elements to be searched. You can specify that a range of elements be searched by coding the THROUGH clause also. This clause is described next:

SEARCH ELEMENT *element-name*

(Required) The name of the action followed by the name(s) of the element(s) you want to search.

The element name can be explicit, partially wildcarded, or fully wildcarded. If you enter a partially wildcarded element name, only those elements matching the criteria are searched.

If you use the THROUGH clause, the element indicated here is the first element in the range to be searched.

THROUGH (THRU) *element-name*

(Optional) Indicates that a range of elements are to be searched, up to and including the element named in this clause. You can use a wildcard with the element name.

The From Clause

The FROM clause identifies the CA Endevor SCM inventory location at which the element search begins. You must enter all FROM information. You need to code the word FROM only once. The FROM clause is described next:

FROM ENVIRONMENT *env-name*

Name of the environment. You must fully specify the environment name; you cannot use a wildcard.

FROM SYSTEM *sys-name*

Name of the system. You can enter a fully specified system name or use a wildcard.

FROM SUBSYSTEM *subsys-name*

Name of the subsystem. You can enter a fully specified subsystem name or use a wildcard.

FROM TYPE *type-name*

Type associated with the element. You can enter a fully specified type or use a wildcard.

Note that you cannot indicate a stage. The search always begins at the entry stage of the environment you specify.

If you specify OPTIONS SEARCH ENVIRONMENT MAP, the utility searches this inventory location first, and then continues the search for the element up the map. If you specify OPTIONS SEARCH ENVIRONMENT ONLY, the utility searches only the environment defined in this clause.

If an element changes as a result of the search, the utility either adds it or updates it at the entry stage of the environment specified in this FROM clause. The element is always added (or updated) at the entry stage of this environment, no matter where the element was retrieved.

The For Clause

The FOR [TEXT] clause identifies the text strings the utility searches for and, optionally, provide the compare ranges to be searched and replacement text for the search strings. If you do not enter column compare range values, the utility uses the COMPARE FROM and COMPARE TO columns associated with the element type.

The IN COLUMNS, BOUNDS ARE, and REPLACE WITH clauses can be entered in any order, as long as they all follow the FOR [TEXT]. These clauses are described in the following: search-string clause.

FOR [TEXT] ***search-string***

Identifies the character string for which the utility will search. This clause is required.

The attributes of the search string are listed as follows:

The minimum length of the search string is 1 character. Null (empty) search strings are prohibited.

- The maximum length of the string is 72 characters. The length of the string must be less than or equal to the number of columns searched (see the description of IN COLUMNS that follows).
- You can specify multiple search strings by repeating the FOR TEXT *search-string* clause. For example:

```
FOR TEXT ABCDE
FOR TEXT BCXYZ
```

- If the string contains imbedded spaces or any other parser delimiter, it must be enclosed by either apostrophes or quotation marks.
  - If the string contains apostrophes, enclose it in quotation marks.
  - If the string contains quotation marks, enclose it in apostrophes.
- The string itself cannot contain both apostrophes and quotation marks.
- Trailing blanks are significant during the search operation if the search-string is quoted.
- The comparison of text strings is case-sensitive; that is, lowercase characters remain lowercase.

[IN] COLUMNS . . .[TO] . . .

Identifies the columns in which the utility looks for the search string (the compare range). This clause is optional.

The values entered here override the compare column values implied by the element type (defined in the FROM clause). The from-column and to-column values must fall within the compare range values (inclusive), however. Otherwise, you receive an error message.

If you do not use this clause, the utility uses the values provided in the BOUNDS ARE clause.

You can enter the following in this clause:

- *from-column* or FIRST, to indicate the first column (inclusive) of the compare range. You can enter any value as the *from-column* value as long as that value is less than or equal to the *to-column* value (see next). If you do not enter a *from-column* but you do enter a to-column, the utility uses the left-column value of the BOUNDS clause as the first column. FIRST reflects the COMPARE FROM value of the type associated with the element.
- *to-column* or LAST, to indicate the ending column (inclusive) of the compare range. The to-column value must be greater than the from-column value, and cannot exceed 32,000. LAST reflects the COMPARE TO value of the type associated with the element.

REPLACE WITH [TEXT] *repl-string*

Identifies the string that will replace the search string in the element. The replacement string has the same attributes as the search string (see the first entry in the FOR clauses table) with the following exception: null (empty) replacement strings are allowed.

This clause is optional. If you use this clause, you need to code the OPTIONS UPDATE ELEMENTS clause (see Options Clauses in this chapter) in order to have the search string replaced and the element(s) updated. Otherwise, the utility runs in search-only mode (and may, depending on what is coded in the execution JCL, generate SCL statements).

If the replacement string is identical to the search string, no elements are searched and an error message is issued.

BOUNDS [ARE] [COLUMNS] . . . [TO] . . .

Identifies--in conjunction with the IN COLUMNS values, if coded--the columns in which the utility looks for the search string. This clause is optional.

If you use this clause, the *left-column* and *right-column* values must be within the range of the element type COMPARE FROM and COMPARE TO columns (inclusive). Otherwise, you receive an error message.

If you do not enter column values, the utility defaults to FIRST and LAST, which reflect the COMPARE FROM and COMPARE TO values of the type associated with the element.

You can enter the following in this clause:

- *left-column* or FIRST, to indicate the first column (inclusive) of the compare range. The utility uses this value if you do not specify an explicit IN COLUMNS *from-column* value. If you do not enter a *left-column* value, the utility defaults to FIRST.
- *right-column* or LAST, to indicate the last column that can be modified in the range. If you do not enter a *right-column* value, the utility defaults to LAST.

Where Clauses

WHERE clauses provide additional element selection criteria. The search is limited to only those elements whose CCID or processor group match the entry in the SEARCH ELEMENT request. WHERE clauses are optional and are described as follows:

WHERE CCID OF . . . EQ . . .

Specifies the CCID that must be associated with the element in order for the utility to search the element. The CCID can be from 1-12 characters in length, and can be explicit, partially wildcarded, or fully wildcarded. Note that coding a fully wildcarded CCID produces the same result as not coding this clause or coding WHERE CCID OF ANY--all elements are selected no matter what the CCID is.

You can provide a list of CCIDs; enclose the list in parentheses. The CCID associated with the element must match at least one CCID in the list to be selected for processing. The utility checks only the current MCF for the CCID.

You can limit the search to only those elements whose base, generate, last action, or retrieve CCID match the CCID specified in the request. Again, the utility checks only the current MCF record.

If you use WHERE CCID OF ANY, all of the above CCID fields in the current MCF record are examined.

WHERE PROCESSOR GROUP EQ . . .

Specifies the processor group that must be associated with the element in order for the utility to search the element. The processor group name can be from 1-8 characters in length, and can be explicit, partially wildcarded, or fully wildcarded. Note that coding a fully wildcarded processor group name produces the same result as not coding this clause--all elements are selected no matter what the processor group is.

You can provide a list of processor groups; enclose the list in parentheses. The processor group associated with the element must match at least one of the processor groups in the list to be selected for processing.

Option Clauses

OPTION clauses allow you to further qualify your request. You can specify none, one, or more than one option. You need to code the word OPTION only once. Option clauses are described next:

CCID [EQ] ***ccid***

Specifies the CCID to be associated with the element when the element is added (or updated) back into CA Endevor SCM. This CCID is assigned to the last action CCID and the generate CCID.

The CCID can be from 1-12 characters in length, and must be explicit.

This clause is optional except under the following condition: the element's system record requires that a CCID be coded and you code the REPLACE WITH clause in the request.

COMMENT [EQ] ***comment***

Specifies the comment to be associated with the element when the element is added (or updated) back into CA Endevor SCM.

The comment can be from 1-40 characters in length. If the comment contains imbedded spaces or punctuation marks, the text must be enclosed by string delimiters.

This clause is optional except under the following condition: the element's system record requires that a comment be coded and you code the REPLACE WITH clause in the request.

BYPASS GENERATE PROCESSOR

Indicates that the generate processor is not to be executed when the element is added back into CA Endevor SCM. By default, the generate processor is invoked whenever an element is added back into CA Endevor SCM.

This clause applies only if you code the REPLACE WITH clause.

PROCESSOR GROUP EQ *group-name*

Assigns a processor group to the element when the element is added back into CA Endevor SCM. The processor group named must exist at the entry stage of the environment.

The processor group can be from 1-8 characters in length, and must be explicit. This clause applies only if you code the REPLACE WITH clause.

OVERRIDE SIGNOUT

If the element is signed out to someone else, override signout must be used and the corresponding authority granted.

You will not get the signout of the element that is fetched. However, the element that is put in the entry stage will be signed out to you.

This clause applies only if you code the REPLACE WITH clause.

SEARCH ENVIRONMENT {MAP | ONLY}

Specifies whether the utility will search beyond the entry stage of the designated environment:

- SEARCH ENVIRONMENT MAP indicates that the utility is to search the environment map for the element if the element is not found at the entry stage of the specified environment.
- SEARCH ENVIRONMENT ONLY indicates that the utility is to search only the environment specified in the SEARCH ELEMENT request. The utility can search both stages of the environment, but not the other environments in the map.

If this clause is not coded, the utility searches only the entry stage of the specified environment. This is the default.

[DATA] TRUNCATION [IS] {PROHIBITED | PERMITTED}

Indicates whether data truncation will take place during string substitution:

- Code DATA TRUNCATION IS PROHIBITED to prevent data in an element record from being truncated. This is the default. An error message is returned if text replacement would have resulted in data truncation.
- Code DATA TRUNCATION IS PERMITTED to allow data in an element record to be truncated. Caution messages are issued in this situation.

LIST DETAILS

Indicates that you want to list, on the Search and Replace Utility Execution Report, each line (or a portion of the line, up to 90 bytes of data) of text containing the search string. The lines are printed as they are encountered during execution of the request. If the request contains a replacement string, the updated line is also printed.

If this clause is not coded, the text of each line containing the text string is not printed in the execution report. Each element searched, along with the number of matches found in that element (0 to 99999), is printed to the execution and summary report, regardless of the LIST DETAILS option setting.

UPDATE [ELEMENTS]

Indicates that the utility is operating in replacement mode. That is, as appropriate, the utility replaces the search string with the replacement string and updates the element. (For more information, see Replacement Mode in this chapter.)

If this clause is coded, you need to code the REPLACE WITH clause if you want to replace the search string and update the element(s). Otherwise, the utility runs in search-only mode.

If this clause is not coded, the utility operates in search-only mode.

Text Replacement

The replacement text string may be equal to, longer, or shorter than the search string. A longer or shorter search string causes data to be shifted to the right or left and blank spaces to be consumed or inserted. The size of the replacement string does not affect the record length, however, except under the following conditions:

- The record has a variable length.
- Extending the record length is required to insert the replacement string.
- The extended length will not exceed the maximum length permitted for the element.

The data record is always padded to its original length.

Note: The search string and the replacement string are case-sensitive.

Compare Column Ranges

By default, the Search and Replace utility searches for the search string in the compare column range associated with the element type definition. You can override the type definition values, however, by specifying values in the IN COLUMNS clause or the BOUNDS ARE clause (or in both) in the SEARCH ELEMENT request.

The BOUNDS ARE clause, used in conjunction with the IN COLUMNS clause, restricts the range of data that is searched and the range of data that may be affected by a change.

- When both the IN COLUMNS and BOUNDS ARE clauses are specified, the utility uses the higher of the IN COLUMNS to-column and the BOUNDS ARE right-column to set the rightmost column that may be affected by data shifting.
- When the IN COLUMNS clause is specified, the search for the search string is limited to the columns indicated.
- When the IN COLUMNS clause is not specified, the utility uses the values specified or implied by the BOUNDS ARE values.
- When the BOUNDS ARE right-column is greater than the IN COLUMNS to-column, data between the to-column and the right-column is usually not changed. The data may be shifted either left or right, depending upon the length of the replacement string. If shifted to the left, the data might be moved into the compare column range. The data then becomes subject to change, as all or part of it could be searched and, possibly, replaced.

  If the replacement string is larger than the search string and data truncation is allowed, some of the “in-between” data may be truncated.

IN COLUMNS Rules

If you use the IN COLUMNS clause, you must follow the rules listed as follows:

- The IN COLUMNS values must be in the range 1 through 32,000, inclusive.
- The IN COLUMNS from-column value must be less than or equal to the to-column value.
- The IN COLUMNS to-column value must be less than or equal to the source element length associated with the element type record.
- The IN COLUMNS from-column and to-column values must be included in the range of FIRST to LAST, respectively. That is, the values must be within the element type COMPARE FROM and COMPARE TO values, inclusively.

- If only a single column number is specified, the utility assumes it represents the from-column value. The to-column is calculated as the from-column plus the size of the search string less 1.
- If only the to-column is specified, the from-column value is assumed to be the left-column value specified or implied in the BOUNDS ARE clause.If both from-column and to-column are omitted, they adopt the values specified or implied by the BOUNDS ARE clause.

BOUNDS ARE Rules

The rules for IN COLUMNS also apply to the BOUNDS ARE clause, with the following exceptions:

- If only a single column number is specified in the BOUNDS ARE clause, the utility assumes it represents the left-column of the range.
- If the left-column value is omitted from the BOUNDS ARE clause, the utility defaults to FIRST. FIRST reflects the COMPARE FROM value of the element type.
- If the right-column value is omitted from the BOUNDS ARE clause, the utility defaults to LAST. LAST reflects the COMPARE TO value of the element type.
- If the BOUNDS ARE left-column does not fall within the IN COLUMNS from-column--to-column range, the IN COLUMNS from-column (or the first column of the element type's compare column range) is used as the beginning column for the search and replace operation.

Shorter Replacement String

When the replacement text string is smaller than the original search string, the original string is replaced and blanks are inserted into the record as follows:

- If no blanks appear between the search string and the rightmost column of the modifiable range, data up to and including that rightmost column is shifted left. Blanks are inserted at the rightmost column of the modifiable range.
- If at least one blank occurs between the text and the rightmost column of the modifiable range but there are no repeating blanks within this range of data, blanks are inserted at the last blank within the range. The data preceding that blank is shifted to the left.
- If the data between the search string and the rightmost column of the modifiable range contains at least two consecutive blanks, additional blanks are inserted at the first occurrence of the repeating blank characters.

Shorter Replacement String Example

The following example illustrates replacement with a text string shorter than the original search string. The x represents a blank space.

```
BOUNDS ARE; 1 and 12
IN COLUMNS: 3 and 10
          ———+----1----+----2
Original text:    AABBBCCCCDDDDDEEExxx
Search string:    BBB
Compare range:    3-10
Replacement string: EE
Updated text:    AAEECCCCDDDxDDEEExxx
```

The modifiable range in this example is 3 through 12. A blank is inserted at column 12 to accommodate for the shorter replacement string. The remaining portion of the original text is not modified at all, as it is outside the range defined.

Longer Replacement String

When the replacement text string is larger than the original string, the search string is replaced and blanks are consumed as follows:

- Data from the original string to the rightmost column of the modifiable range is searched from left to right. All repeating blank characters are consumed as required to perform the substitution. When the appropriate number of blanks have been consumed, the data between the end of the text string and the rightmost column of the modifiable range is shifted to the right and the replacement string is inserted into the record.
- If there are not enough extra blank characters and data truncation is permitted (specified in the request), the utility performs data truncation. The data at the rightmost column of the modifiable range is deleted, as necessary, to provide space for the replacement string. The utility issues a cautionary message and continues processing the element.

  If there are not enough blank spaces for the replacement string and data truncation is prohibited, the utility does the following:

  Generates an error message.

  - Displays the data for the element on the Search and Replace Utility Execution Report.
  - Continues to search the element for other search and replace operations.
  - Does not update the element.

Longer Replacement String Examples

The following examples illustrate text replacement with a string longer than the original search string. The x represents a blank space.

Example 1

```
BOUNDS ARE: 1 and 26
IN COLUMNS: 3 and 10
           ----+----1----+----2...
Original text:    AABBBCCCCDDDDDxxx
Search string:    BBB
Compare range:    3-10
Replacement string: EEEEEE
Updated text:    AAEEEEEECCCCDDDDD
```

The modifiable range in this example is 3 through 26. The replacement string fits within the compare columns and is only three characters longer than the search text. Consequently, the three blanks at the end of the original text are consumed by the replacement string as the data shifts to the right.

Example 2

```
BOUNDS ARE: 1 and 10
IN COLUMNS: 3 and 10
            ----+----1----+----2...
Original text:     AABBBCCCCDDDDDxxx
Search string:     BBB
Compare range:     3-10
Replacement string:  EEEEEE
Truncation permitted: AAEEEEEECDDDDDxxx
Truncation prohibited: Error
```

The modifiable range in this example is 3 through 10. The replacement string is too long to replace only the search string within the modifiable range. If truncation is permitted, the utility replaces the search string, and the characters following the search string up through column 10, with the replacement string. The remainder of the original text, starting in column 11, is not modified.

If truncation is not permitted, you receive an error message and processing stops for this element.

Example 3

```
BOUNDS ARE: 1 and 23
IN COLUMNS: 3 and 10
          ———+----1----+----2...
Original text:    AABBBxxCCxxDDDDxxxGGG
Search string:    BBB
Compare range:    3-10
Replacement string: EEEEEEE
Updated text:     AAEEEEEEExCCxDDDDxGGG
```

The modifiable range in this example is 3 through 23. The replacement string can replace the search string and shift data to the right, within the modifiable range. Repeating blanks are consumed in such a way as to shift the remainder of the original string to the right--the replacement string is four characters longer than the search string so four blanks were consumed.

Multiple Occurrences of the Search String

The data record may contain multiple occurrences of the search string. The scan for subsequent appearances of the string begins immediately after the last character of the replacement string in the modified record. See the following example:

```
Original text:    ABCBE
Search string:    B
Replacement string: QQ
Updated text:     AQQCBE
Final result:     AQQCQQE
```

The first occurrence of B is replaced by QQ. The utility begins its search for the next appearance of B after the second Q in the text string.

Note: The results of search and replace with multiple occurrences of the search string may not be what you expect, due to data being shifted into and out of the compare column range by replacement strings.

If you had a compare column range of 1-4 in the previous example, the utility would not replace the second occurrence of B, because the search string is now outside the specified column range. Similarly, if a replacement string is shorter than the search string, data may move into the compare column range that would not otherwise be included in the search.

Search and Replace Utility Reports

The Search and Replace utility generates three reports as part of its normal processing:

Control Statement Summary

Shows the control statements that were provided in the ENSSCLIN DD statement and identifies any parser or statement validation errors.

Execution Report

Contains information about the execution of each request.

Summary Report

Summarizes each request processed. The summary indicates the element name, the return code, the number of matches found, the location where the match was found, and the location where the element was added (updated) back into CA Endevor SCM.

These reports are written to the ENSMSGS1 DD statement.

The Search and Replace Control Statement Summary Report

The Search and Replace Control Statement Summary Report shows the control statements coded in the ENSSCLIN DD statement, and whether there are any parser and validation errors.

The Search and Replace Utility Execution Report

The column ruler is printed once for each element containing a search string. The ruler may be reprinted if a subsequent search string is found in the element but cannot be displayed using the column ruler shown; for example, the starting column number is not valid for the second search string. In this situation, a new ruler, with the appropriate starting column number, is printed for the element.

The Search and Replace Utility Summary Report

The Search and Replace Utility Summary Report provides the following information for each request processed:

Statement Number

The statement number associated with a SEARCH ELEMENT request.

If the SEARCH ELEMENT request contains multiple FOR TEXT clauses and at least one search string was found in the element, multiple report lines are generated for the element. The first line provides the element information, and lists the **total** number of matches found. Each additional line for the element (numbered nn.1, nn.2...nn.x, for x number of FOR TEXT clauses) lists the number of matches found for a **specific** FOR TEXT clause.

Page Number

The page number on the Search and Replace Utility Execution Report at which processing for the element began.

Element

The name of the CA Endevor SCM element that was processed.

Return Code

The return code associated with the element's SEARCH ELEMENT request.

Lines Searched

The number of times the FOR TEXT clauses are searched for the FOR TEXT string.

Matches Found

The number of times the FOR TEXT strings were found. This value represents the actual number of occurrences, not the number of lines that contain the search string.

Location Where Found

The inventory location from which the element was retrieved.

Location of Add/Update Operation

The inventory location to which the element was added or updated. This field is blank if the element did not contain the search string or if the utility is running in search-only mode.

Text Search and Replace Usage Scenarios

This section contains usage scenarios to show you how the Search and Replace utility performs in various situations. Three scenarios are presented:

- Scenario 1--A simple search without any options.
- Scenario 2--A simple search with replacement text, in search-only mode.
- Scenario 3--A search in replacement mode, using the SEARCH ENVIRONMENT MAP feature, multiple FOR TEXT clauses, and the BOUNDS ARE parameter.

Introduction

Three elements constitute the sample elements for demonstrating the effects of the SEARCH ELEMENT SCL statement and its optional parameters. The sample elements each represent a different type: C source code (type C), COBOL source code (type COB), and text (type TXT). To highlight the SEARCH ENVIRONMENT option, the type C and type COBOL elements are stored in Stages A and B, respectively, in the first environment. The type TXT element is stored at Stage D of the second environment. Stage A in this example is the entry stage for the first environment. Entry stages are defined through the C1DEFLTS environment table.

- When the OPTIONS SEARCH ENVIRONMENT clause is not coded, the utility searches only Stage A of the first environment. This happens in Scenario 1.
- When the OPTIONS SEARCH ENVIRONMENT ONLY clause is specified, the utility searches both Stages A and B of the first environment. This happens in Scenario 2.
- When the OPTIONS SEARCH ENVIRONMENT MAP clause is specified, the entire map is searched. This happens in Scenario 3.

The Test Elements

The utility is executed against three elements:

- HELLO.C, which is a C program
- HELLO.COB, which is a COBOL program
- HELLO.TXT, which is a text file that further explains (sets) the scene

The elements are shown in the following sections.

The results of the search depend upon the attributes of each element as well as the information provided in the SEARCH ELEMENT request.

The HELLO.C Element

Element HELLO.C

```
#include <stdio.h>

void main()
{
  printf("Hello, world!\n");     /* print 'Hello' message */
}
```

The HELLO.COB Element

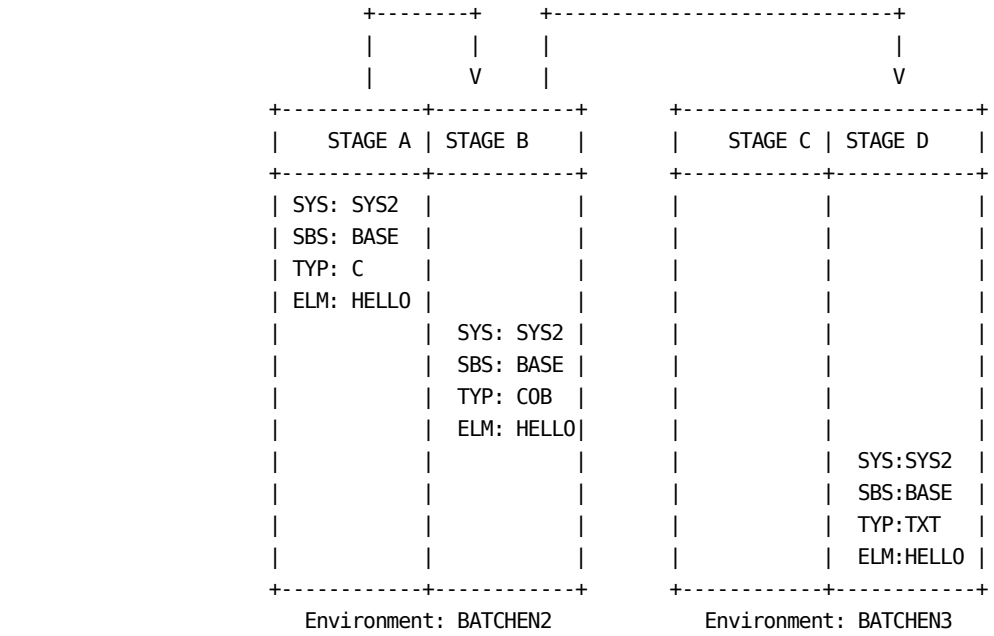
Element HELLO.COB

```
000100 ID DIVISION.
000200 PROGRAM-ID.   HELLO.
000300 AUTHOR.       DEVELOPMENT.
000400 INSTALLATION.  CA.
000500 DATE-WRITTEN.  FEBRUARY 14, 1994.
000600 DATE-COMPILED.
000700*************************************************************
000800*  TRIVIAL PROGRAM TO DISPLAY 'HELLO' MESSAGE               *
000900*************************************************************
001000 SKIP3
001100 ENVIRONMENT DIVISION.
001200 CONFIGURATION SECTION.
001300 SOURCE-COMPUTER. IBM.
001400 OBJECT-COMPUTER. IBM.
001500 INPUT-OUTPUT SECTION.
001600 FILE-CONTROL.
001700**************
001800 DATA DIVISION.
001900**************
002000************************************************
002100 FILE SECTION.
002200************************************************
002300************************************************
002400 WORKING-STORAGE SECTION.
002500************************
003000*****************************************************************
003100*****************************************************************
003200*******************
003300 PROCEDURE DIVISION.
003400*******************
003500   SKIP1
003900   DISPLAY 'HELLO, WORLD!'
004500   GOBACK.
```

The HELLO.TXT Element

Element HELLO.TXT

Three elements, all named HELLO, comprise the sample elements for demonstrating the effects of the SEARCH ELEMENT SCL statement and its optional parameters. The sample elements are of type C, COB and TXT representing C source code, COBOL source code and this text document. To further highlight the SEARCH ENVIRONMENT option, the type C and COBOL elements will be stored in stages A and B, respectively, in the first environment and the type TXT element will be stored at stage D of the second environment; the environment mapping and the element location established for the sample reports is as follows:



When OPTIONS SEARCH ENVIRONMENT is omitted, only stage A, the entry stage of the environment, is searched. When OPTIONS SEARCH ENVIRONMENT ONLY is specified, both stages A and B are searched, because A is the entry stage.
(Note: If B is the entry stage, only B is searched.)

Finally, when OPTIONS SEARCH ENVIRONMENT MAP is specified, the entire map is searched. In all cases, the first occurrence of an element is processed.

Scenario 1: Simple Search in Search-Only Mode

Scenario 1 is a simple search and demonstrates what happens when the SEARCH ELEMENT request contains no options, including the SEARCH ENVIRONMENT option. When the SEARCH ENVIRONMENT option is omitted, the utility searches only the entry stage of the specified environment. In this scenario, then, the utility searches only Stage A of environment BATCHEN2.

Simple Search SCL

The SCL for this request is shown here:

```
SEARCH ELEMENT HELLO
 FROM ENVIRONMENT BATCHEN2
      SYSTEM SYS2
     SUBSYSTEM BASE
       TYPE *
 FOR TEXT 'i'
```

Simple Search Output

The output from processing this request appears in the following sections.

The Search and Replace Control Statement Summary Report

No syntax or validation errors occurred. Processing continues for this request.

The Search and Replace Utility Execution Report

Only one element--HELLO.C--is searched for matches. The utility finds six matches for the FOR TEXT string in this element.

As mentioned above, the Search and Replace utility found 6 matches of the FOR TEXT string. The location in which the matches were found is listed.

Note that there is no entry in the LOCATION OF ADD/UPDATE OPERATION field. This is because the utility is operating in search-only mode. The element is not changed and not updated.

Scenario 2: Simple Search with Replace in Search-Only Mode

Scenario 2 is a simple search with replacement in search-only mode, demonstrating what happens in two situations:

- You do not code the OPTIONS UPDATE ELEMENTS clause, but you do include a replacement string in the SCL.

  In this scenario, assume the ENSSCLOT DD statement has been allocated in the execution JCL. SEARCH ELEMENT SCL is generated for the element containing the search string.

- You code OPTIONS SEARCH ENVIRONMENT ONLY.

  In this scenario, the utility searches both Stage A and Stage B of environment BATCHEN2.

Simple Search with Replace Mode SCL

The SCL for this request is shown here:

```
SEARCH ELEMENT HELLO
 FROM ENVIRONMENT BATCHEN2
      SYSTEM SYS2
     SUBSYSTEM BASE
       TYPE *
 FOR TEXT 'i'
   REPLACE WITH '<i>'
 OPTIONS CCID = "CCID-99"
     COMMENT = "Test scenario number 2"
     SEARCH ENVIRONMENT ONLY
     LIST DETAILS.
```

Simple Search with Replace Output

The utility generates SCL for any elements containing the search string, because a replacement string was provided in the SEARCH ELEMENT request and the ENSSCLOT DD statement was allocated. This SCL can be used as input when you run the utility again.

The utility also produces syntax, execution, and summary reports.

No syntax or validation errors occurred. Processing continues for this request.

The utility searched two elements, HELLO.C and HELLO.COB, but only element HELLO.C contains a match for the search string. Element HELLO.COB does contain the letter I, but not in lowercase format. Because the search string is case-sensitive, the utility does not consider the text a match.

The Search and Replace utility found three matches in element HELLO.C and no matches in HELLO.COB. The report lists the location where both elements were found. Note that the utility searched both Stage A and Stage B of environment BATCHEN2.

There is an entry in the LOCATION OF ADD/UPDATE OPERATION field, for element HELLO.C. This entry is for reference purposes only. The element is not modified and added or updated back into CA Endevor SCM because the OPTIONS UPDATE ELEMENTS clause was not coded in the SEARCH ELEMENT request. The entry is provided to let you know what location will be affected should you decide to update the element with the replacement string.

The generated SCL statement is shown here:

```
SEARCH ELEMENT 'HELLO'
 FROM ENVIRONMENT 'BATCHEN2'
    SYSTEM 'SYS2'
    SUBSYSTEM 'BASE'
    TYPE 'C'
 FOR TEXT 'i'
   REPLACE WITH TEXT '<i>'
 OPTIONS CCID = 'CCID-99'
     COMMENT = 'Test scenario number 2'
     SEARCH ENVIRONMENT ONLY
     DATA TRUNCATION IS PROHIBITED
     LIST DETAILS
     UPDATE ELEMENTS
.
```

Scenario 3: Search Environment Map, Replace, and Update

Scenario 3 is a search and replace operation, with several options coded to limit the search. This scenario demonstrates processing with the following:

- The SEARCH ENVIRONMENT MAP feature
- Multiple FOR TEXT clauses
- The BOUNDS ARE clause (for one FOR TEXT clause)
- The WHERE CCID clause
- The LIST DETAILS clause

Search Environment Map SCL

The SCL for this request is shown here:

```
SEARCH ELEMENT HELLO
 FROM ENVIRONMENT BATCHEN2
      SYSTEM SYS2
     SUBSYSTEM BASE
       TYPE *
 FOR TEXT 'i' REPLACE WITH '*'
 FOR TEXT 'T' REPLACE WITH TEXT '$'  BOUNDS ARE 1 TO LAST
 WHERE CCID = 'CCID-02'
 OPTIONS CCID = "CCID-99"
     COMMENT = "Test scenario number 3"
     SEARCH ENVIRONMENT MAP
     LIST DETAILS
     UPDATE ELEMENTS.
```

BOUNDS ARE is set to BOUNDS ARE 1 TO LAST to demonstrate type compare column checking in relation to the SCL statements.

WHERE CCID limits the search to those elements whose CCID is CCID-02.

The CCID and comment provided in the OPTIONS clause are assigned to the element when it is added (updated) back into CA Endevor SCM.

LIST DETAILS tells the utility to print the original line of text where the search string is found and the line of text after the string is replaced.

UPDATE ELEMENTS indicates that the element is to be added or updated into CA Endevor SCM after the text string is replaced.

Search Environment Map Output

The output from processing this request is shown next. Along with the syntax, execution, and summary reports is a copy of the updated element.

No syntax or validation errors occurred. Processing continues for this request.

The utility searched only one element--HELLO.TXT. The element was found up the map, in Stage D of BATCHEN3. Substitution was performed, as appropriate, for each FOR TEXT clause.

Note message ENBS048E (immediately after the SCL request). A matching element was found in Stage A of environment BATCHEN2, but the column values specified in the SEARCH ELEMENT request are outside the compare column values for the element. This is why only one element was searched.

Note the information message lines ENBS016I and ENBS017I. ENBS016I lines present the original line of text containing the search string. ENBS017I lines show the line of text after the search string has been replaced.

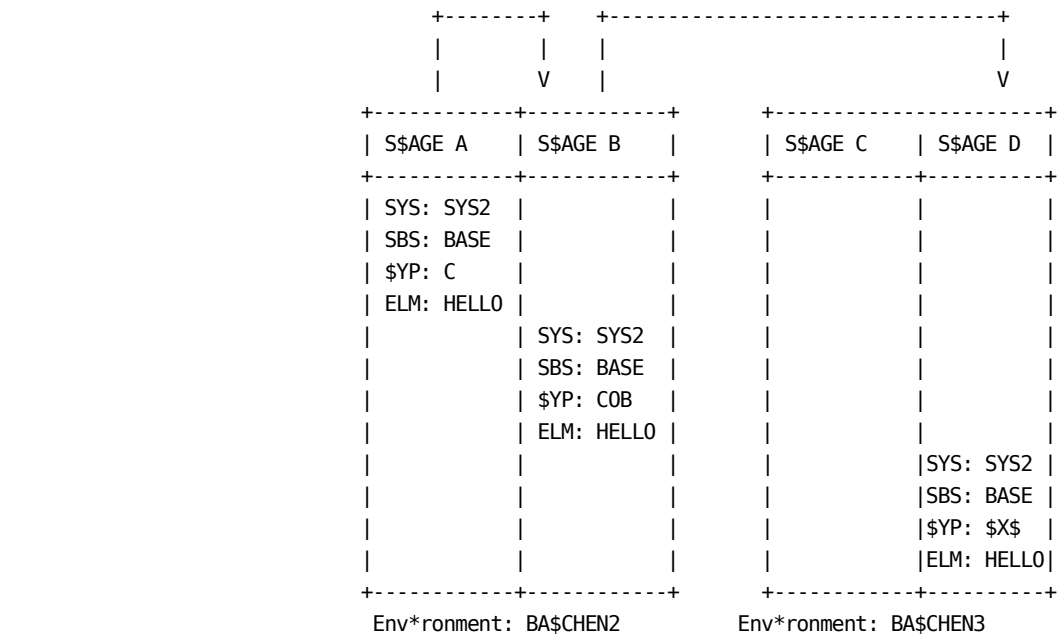
A processing code of 12 was returned for the request. The utility processes a request as long as the return code does not exceed 12.

The Search and Replace utility found a total of 66 matches for the search strings in the SEARCH ELEMENT request. All information for the element is presented on the first line for Statement 1. Statement line 1.1 indicates the number of matches found for the first FOR TEXT clause. Statement line 1.2 indicates the number of matches found for the second FOR TEXT clause.

Although the element was found in Stage D of environment BATCHEN3, it is added into CA Endevor SCM at the entry stage (Stage A) of the base environment (BATCHEN2).

The result of the replace operation on element HELLO.TXT:

```
Three elements, all named HELLO, compr*se the sample elements for
demonstrat*ng the effects of the SEARCH ELEMEN$S SCL statement and
*ts opt*onal parameters. $he sample elements are of type C, COB and
$X$ represent*ng C source code, COBOL source code and th*s text
document. $o further h*ghl*ght the SEARCH ENVIRONMEN$ opt*on, the
type C and COBOL elements w*ll be stored *n stages A and B,
respect*vely, *n the f*rst env*ronment and the type $X$ element w*ll
be stored at stage D of the second env*ronment; the env*ronment
mapp*ng and the element locat*on establ*shed for the
sample reports *s as follows:
```




```
When OP$IONS SEARCH ENVIRONMEN$ *s om*tted, only the entry stage of

the env*ronment, *s searched. When OP$IONS SEARCH ENVIRONMEN$ ONLY *s
spec*f*ed, both stages A and B of the entry env*ronment are searched.
F*nally, when OP$IONS SEARCH ENVIRONMEN$ MAP *s spec*f*ed, the ent*re
map *s searched. In all cases, the f*rst occurrence of an element
*s processed.
```

Update Output Data Set Name in Component List

You can use the Search and Replace utility to update an element's component list to change the output data set name. This allows you to update the data set name to reflect the current location of the output, which became outdated when the element was moved and not regenerated. This data is updated in place. For example, if the element is at stage 2, it is not fetched back to stage 1, which would be done if the utility was used to update element source.

You might want to use this option to address problems that can occur when the component data contains out of date information. For example, an error occurs when using LL (List Listing) from the Quick Edit or List Element panels for an element that was moved but not regenerated. In that case, the component data points to the list lib used when the element was last generated, not to the current list lib in which the listing resides. This causes the LL to fail because the member is no longer present in that library. The same problem can exist for other output libraries (such as LOADLIBs) and automated CONDELE with 'PARM=*COMPONENTS'.

Any user with ESI security move authority for the element is authorized to update the component data at stage 2, if it is a non-entry stage. Any user with ESI security update authority for the element is authorized to update the component data at stage 1, even if stage 1 is a non-entry stage. As with element source changes, a history of component data changes is maintained. Therefore, a user can see who changed output component data and when the change was made.

The JCL used to execute this utility can be found in member ENBSRPL1, in the JCL library iprfx.iqual.CSIQJCL. To specify the changes you want, you need to code an SCL statement in the JCL. The SCL specifies what component lists will be searched, what data set names will be searched for, and what data set names they will be replaced with.

In addition to replacement mode, you have the option to run the utility in validate mode to ensure the SCL is properly coded, or in search-only mode, which produces a report showing the component lists that include the data set names that are the object of your search.

To update an output data set name in a component list

1. Add a valid job card at the front of the ENBSRPL1 job stream.
2. Change the data set names and other variables in the JCL to the appropriate values for your installation.

3. Code a Search Element Component List statement in the JCL's ENSSCLIN DD statement, as appropriate for the action you want to perform.

   To search for and replace a data set name, code the For clause and the Replace With clause and run the utility in the replacement mode by including the Options clause with the Update option. To search for and replace more than one data set name, you can include multiple combinations of the For clause and Replace With clause in one Search Element Component List statement.

   Note: Before you update the component list, you may want to produce a report that lists the data sets found by the search. To produce a report instead of updating the component list, make sure that your SCL does **not** include the Update option on the Options clause. To produce a report, code at least the required keywords and variables of the Search Element Component List statement, which includes specifying the complete location of the element and the FOR clause to specify the data set you are searching for. However, do not include the Options clause Update option. To search for more than one data set name, you can include more than one For clause in one Search Element Component List statement.

   For details, see Search Element Component List (see page 211).

   The SCL statement you code is used to specify what action the utility performs.

4. (Optional) Add the validate parameter in the execution JCL after the PARM=ENBS1000 statement:

   ```
   PARM='ENBS1000VALIDATE'
   ```

   You can abbreviate VALIDATE in the parameter, using any of the following entries: V, VA, VAL, VALID, VALIDA, or VALIDAT

   The utility will operate in validate mode.

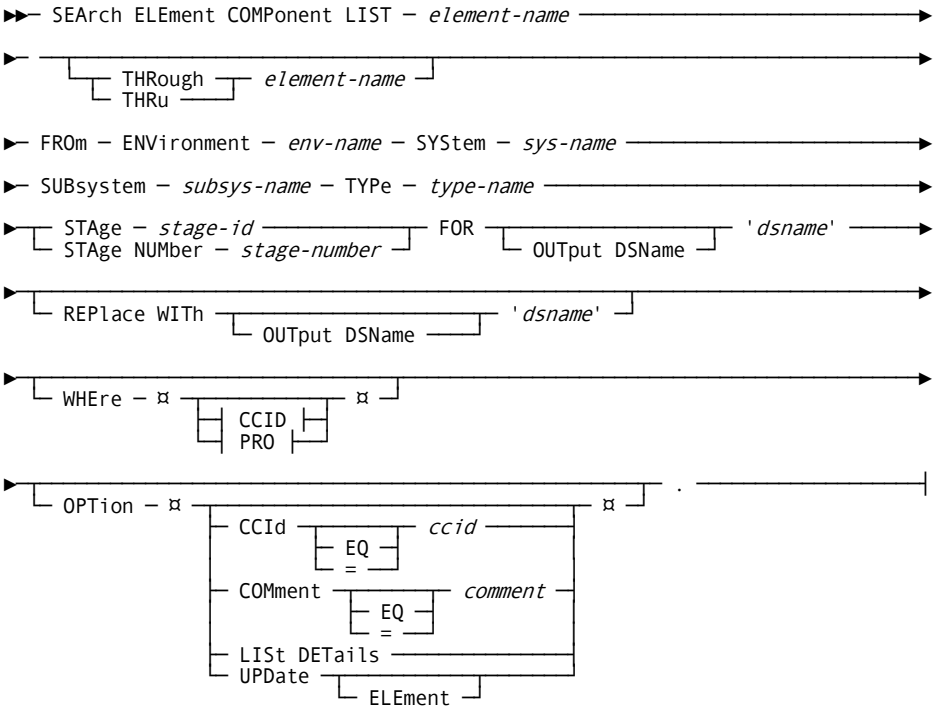
5. Execute the Search and Replace utility.

   The results of the Search and Replace utility depend on the how you coded the Search Element Component List statement and whether you included the validate parameter on the execution JCL.

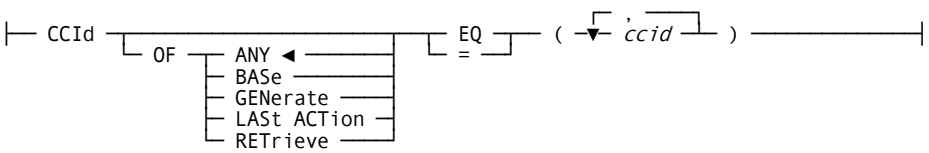
   - If you specified the validate parameter, then the utility parses and verifies all the requests in the generated SCL statements for proper syntax, but does not execute the requests even if no syntax or validation errors are found. If no errors are found, the SCL statements are formatted, but processing stops after validation. The SCL actions are not performed. Errors other than syntax errors are not noted at this time.
   - If you did **not** specify the validate parameter, then the utility parses and validates all the requests before processing them. If the parser or validation routine detects an error, the utility will not execute any of the statements. The parsing routine attempts to parse all of the control statements before terminating. If no errors are found, then the SCL action is performed.
     - If the SCL does **not** include the Update option, then a report is produced.
     - If the SCL does include the Update option, then the component list's output data set name that is the object of the search (as specified in the FOR clause) is replaced with the data set name (specified in the Replace clause).

Search Element Component List

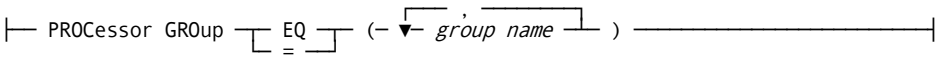
The Search Element Component List syntax for the Search and Replace utility, ENBSRPL1, is shown next:



Expansion of CCID



Expansion of PRO



Parameters

This action uses the following parameters:

SEArch ELEment COMPonent LISt *element*

Specifies one or more element component lists to be searched.

THRough *element*

THRu *element*

(Optional) Indicates that a range of elements are to be searched, up to and including the element named in this clause. The element in the Search Element Component List clause specifies the first element in the range.

FROM

Identifies the inventory location from which the search begins. You must use the keywords environment, system, subsystem, type, and either the stage or the stage number.

ENVironment *env-name*

Identifies the environment. The environment must be fully specified, you cannot use a name mask.

SYStem *sys-name*

Identifies the system. You can fully specify the name or use name mask.

SUBsystem *subsys-name*

Identifies the subsytem. You can fully specify the name or use name mask.

TYPe *type-name*

Identifies the type. You can fully specify the name or use name mask.

STAge *stage-id*

STAge NUMber *stage-number*

Identifies the stage. You can use either the stage or the stage number, but not both. You must fully specify the stage or the stage number; name masking is not allowed. This value is the stage at which the element exists. Component data will be updated in place, that is there will be no copy back from stage 2 to stage 1.

FOR *dsname*

FOR OUTput DSName *dsname*

Identifies the name of the data set that is the object of the search. You can specify FOR *dsname* or FOR OUTput DSName *dsname*; the OUTput DSName keyword is optional. Name masking is not allowed.

REPlace WITh *dsname*

REPlace WITh OUTput DSnam *dsname*

(Optional) Specifies a data set name to replace the data set name identified in the FOR clause. You can specify REPlace WIth *dsname* or REPlace WIth OUTput DSName *dsname*; the OUTput DSName keyword is optional. The value must reference an existing data set name. Name masking is not allowed.

If this option is not specified, then the utility will simply search for the output data set component name identified in the For clause.

You can include more than one combination of For clause and Replace With clauses in a single Search Element action. The data set specified in the For clause is replaced with the data set specified in the immediately following Replace clause.

If an invalid data set name is specified or an uncataloged Replace With data set name is specified, then none of the clauses will be processed.

WHEre

(Optional) Specifies additional element selection criteria. Limits the search to only those elements whose CCID or processors group match the specified value.

CCID OF ANY|BASe|GENerate|LASt ACTion|RETrieve EQ|= *ccid*

Specifies the CCID that must be associated with the element for the utility to search the element. You can use either the keyword EQ or the equal symbol (=) in this clause.The *ccid* value can be 1 to 12 characters. Name masking is allowed. If you fully wildcard the *ccid* value, or you do not specify this clause, elements are selected regardless of their CCID.

You can specify a list of CCIDs, with each value separated by a comma and the entire list enclosed in parentheses. For an element to be selected for processing, the CCID associated with the element must match at least one CCID in the list. The utility checks only the current MCF for the CCID.

OF ANY|BASe|GENerate|LASt ACTion|RETrieve

(Optional) Specifies where the matching CCID can be located to limit the search to only those elements whose base, generate, last action, or retrieve CCID match the CCID specified in the request.

ANY

(Optional) All CCIDs in the current MCF are searched. The default.

BASe

(Optional) All CCIDs in the current MCF associated with a base level element are searched.

GENerate

(Optional) All CCIDs in the current MCF associated with a generate action on an element are searched.

LASt ACTion

(Optional) All CCIDs in the current MCF associated with the last action on an element are searched.

RETrieve

(Optional) All CCIDs in the current MCF associated with a retrieve action on an element are searched.

PROCessor GROup EQ|= *group*

Specifies a 1- to 8-character processor group that must be associated with the element in order for the utility to search the element. Name masking is allowed. You can specify a list of processor groups, with each value separated by a comma and the entire list enclosed in parentheses. For an element to be selected for processing, the processor group associated with the element must match at least one of the processor groups in the list.

OPTions

(Optional) The options clause allows you to further qualify your request. You can specify none, one, or more than one option. The CCID and COMMENT options maybe required depending on your element system record requirements. The Update Element option is required to run the utility in replacement mode.

CCID EQ|= *ccid*

(Optional) Specifies the CCID to be associated with the element when the element is added (or updated) back into CA Endevor SCM. This CCID is assigned to the last action CCID and the generate CCID. The CCID can be from 1 to12 characters in length, and must be explicit. This clause is optional except under the following condition: the element's system record requires that a CCID be coded and you code the REPLACE WITH clause in the request. It is optional to include either the EQ keyword or the equal sign (=), neighther are required.

COMMENT EQ|= *comment*

(Optional) Specifies the comment to be associated with the element when the element is added (or updated) back into CA Endevor SCM. The comment can be from 1-40 characters in length. If the comment contains imbedded spaces or punctuation marks, the text must be enclosed by string delimiters. This clause is optional except under the following condition: the element's system record requires that a comment be coded and you code the REPLACE WITH clause in the request.

LISt DETails

(Optional) Indicates that you want to list, on the Search and Replace Utility Execution Report, each output data set name component found and, if run in replace mode, each new output data set name. Each element searched, along with the number of matches found in that element (0 to 99999), is printed to the execution and summary report, regardless of the LIST DETAILS option setting.

UPDate ELEment

(Optional) Indicates that the utility is operating in replacement mode. The utility replaces the data set name specified in the For clause with the data set name specified in the Replace clause. The utility operates in search-only mode, if either the Replace With clause or the Update Element clause is not coded.

Chapter 12: Using the Unload, Reload, and Validate Utility

This section contains the following topics:

The Unload, Reload, and Validate Utility

The Unload, Reload, and Validate utility (program C1BM5000) is a backup, recovery, and file validation mechanism for CA Endevor SCM VSAM control files (Master Control File, package data sets) and their related base and delta libraries. It allows users to backup (Unload), restore (Reload), and/or validate (Validate) the integrity of one or more CA Endevor SCM environments in the event of a physical device failure or site disaster.

The Unload, Reload, and Validate utility provides a point-in-time physical recovery mechanism. The utility can be used to perform the following actions:

- Unload all or specific environments and/or systems and their related elements/components.
- Reload all or specific environments and/or systems and their related elements/components.
- Validate the integrity of all or specific environments and/or systems and their related elements/components.

The Unload, Reload, and Validate utility provides a point-in-time physical recovery mechanism. It allows you to backup (unload), restore (reload), and validate the integrity of one or more CA Endevor SCM environments in the event of a physical device failure or site disaster.

The Unload function copies all Master Control File environment information (system, subsystem, type, type sequence, data set, and element master record) to the data set you specify. Then it copies all the elements in either full or incremental mode. If an incremental unload is specified on the Unload request, then base and delta information is only copied for elements that have changed since the last unload. During the unload process, validation always occurs to ensure that the elements to be unloaded meet certain integrity criteria. Any corrupt elements are not unloaded. You can run the Validate function independent of the Unload function.

The Reload function restores data from the data sets created by the unload process. This chapter provides examples of how you can use the Reload function to recover from hardware failure.

This chapter discusses the Unload, Reload, and Validate functions performed by program C1BM5000. For each, there is a brief description of the function, a syntax diagram, and a discussion of the rules that each will follow.

Note: For the element catalog, the Reload utility uses normal CA Endevor SCM inventory management functions which will update the MCF, BASE/DELTA and element catalog and EINDEX where necessary.

ACMQ Files and the Reload Utility

The Reload utility does not update the ACMQ files. The ROOT and XREF files remain at the state that they were in before the reload. ACMQ data is not touched by the Reload utility, but some forms of recovery may require a full or partial rebuild of ACMQ data.

How the Reload utility affects the validity of the ACMQ files in various situations is described next:

- Suppose you lose a delta and decide to reload the missing pieces.

  In this case the ROOT and XREF files before the reload will contain correct data and thus the reload will not impact the quality of the content.

- Suppose you lose the ACMQ files.

  In this case reload will not help you. You could restore the data from another type of backup (full volume, IDCAMS), but the ACMQ data could be out-of-sync with the MCF and Catalog data. For this case, you will have to rebuild the ACMQ data sets (BC1JACML)

- Suppose you want to recover some elements that have been deleted from a full or incremental unload.

  In this case, we recommend using Transfer from unload and regenerate the elements. The Transfer action will rebuild the outputs and the ACMQ data. If you want to use Reload, because you do not want to rebuild the outputs, then ACMQ will be out of sync and you will need to run a partial rebuild of the ACMQ data for these reloaded elements.

The Unload Function

The Unload function unloads and validates the contents of the VSAM Master Control Files (MCFs), base and delta files associated with the environments and systems specified on the job request. The file created by the Unload function contains a backup of all internal MCF definitions (system, subsystem, type, type sequence, data set, element master record) and base/delta data (element base, element delta, component base, component delta). Packages contained within a package data set can also be unloaded.

Unload may be run for an entire environment, or for selected systems within an environment. Unload may also be directed to backup an entire package data set or individual packages.

Note: The LRECL of the unload data set must be at least 84 bytes larger than the largest type record length in the unloaded environment/system.

Regardless of whether you specify a full or incremental unload, all Master Control File environment information (system, subsystem, type, type sequence, data set, and element master record) will be unloaded to the data set you specify. This is done so that the environment definitions can be restored from any unload point.

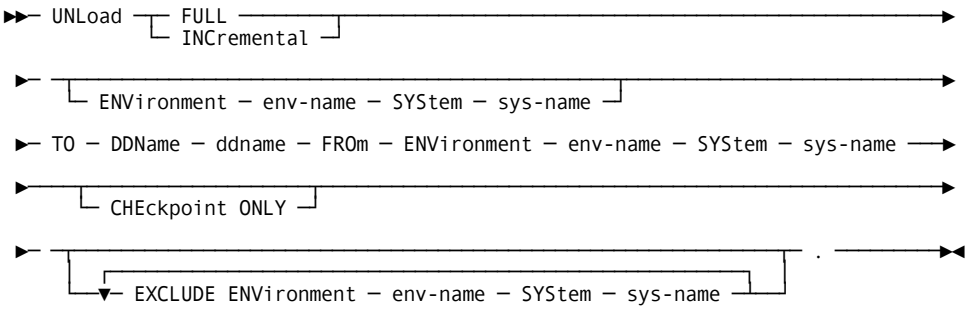
After unloading the environment information, Unload then unloads elements in either a full or incremental mode.

This unloading of environment information and elements occurs in the following order:

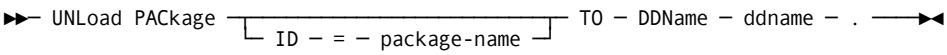
1. Stage 1 internal environment definitions.
2. Stage 2 internal environment definitions.
3. Stage 1 elements.
4. Stage 2 elements.

Unload Control Card

The following syntax provides the parameters for using Unload against an environment or system:



The following syntax provides the parameters for using Unload against one or more package data sets:



The parameters for the Unload control card are described in the following:

UNLOAD

Specifies the UNLOAD function. You must further qualify the UNLOAD request with one of the following:

- FULL-Unloads Master Control File environment and related base/delta information for all elements in the environment(s) and system(s) specified in the FROM statement.
- INCREMENTAL-Unloads Master Control File information for all elements in the environment(s) and system(s) specified in the FROM statement. Base and delta information will only be unloaded for elements that have changed since the last unload, based on the date/time stamp for each system. Any number of UNLOAD statements may be coded for a single run.

Note: Any change to an element (SOURCE, MCF fields or STATUS (that is, SIGNED IN)) is considered and the unload incremental occurs for these changes.

TO DDNAME

Identifies the DDname to which the unload data set will be assigned for both environment and package processing.

Note: If you specify a DDname, you cannot use the CHECKPOINT ONLY clause.

FROM

Allows you to specify the FROM location for a combination of environments and systems to be unloaded. The qualifying statements are:

- ENVIRONMENT-Identifies the environment from which information is to be unloaded. A name mask may be used.
- SYSTEM-Identifies the system from which information is to be unloaded. A name mask may be used.

CHECKPOINT ONLY

Causes a FULL UNLOAD request to simply update the system backup time stamp so you can use something other than the Unload utility to do the full backup and use the Unload utility for incremental backups. You must specify both the FULL and the FROM clauses when using this option.

Note: If you specify CHECKPOINT ONLY, you cannot specify a DDname.

EXCLUDE

Bypasses unload processing for the specified environments and systems that are a subset of the environments and systems to be unloaded. CA Endevor SCM does issue an Enqueue for the environment and system combinations specified on the Exclude clause. You can specify multiple Exclude clauses. Wildcards and placeholders are allowed. For example:

- If you specify UNLOAD FULL ENV '*' SYS '*' but do not want to process environment ENVTEST system SYSTEST, you would specify:

  ```
  UNLOAD FULL FROM ENV '* ' SYS '*' TO DDNAME UNLODNN EXCLUDE ENV 'ENVTEST' SYS 'SYSTEST'
  ```

- If you specify more than one Unload SCL statement and they are identical with the exception of the Exclude clause, then the Exclude clause from the last SCL is the one used.

- If you specify more than one Unload SCL statement and the they are identical except that the Exclude clauses are different and the environment and system combinations to be unloaded overlap, then the most recent Exclude statements apply to the specified environment and systems to be unloaded. For example:

  ```
  UNLOAD FULL FROM ENV '*' SYS '*' EXCLUDE ENV 'ENVT*' SYS 'SYST*' TO UNLOAD UNLODNN.
  ```

  In this case, when any environment whose name begins with P is about to be unloaded, then any environment with the name PROD and the system name SYSX is excluded. In all other cases, the prior Exclude clause takes effect.

PACKAGE

Required when unloading one or more packages. You may also select a package identifier (ID) to identify the name of a specific package to be unloaded. You can use a name mask. If you do not specify a package ID, all packages in the package data will be unloaded.

For promotion packages, you cannot specify the package ID of individual historic versions. You can specify the fully qualified name or mask package ID of the current version. This unloads the records for all versions of a promotion package.

Full Unloads

All base and delta members and related component list base and delta members are unloaded for each element associated with the environments and systems specified in the UNLOAD request.

Incremental Unloads

Only elements associated with the environments and systems specified in the Unload request that have changed since the last unload will be unloaded.

When Unload is successfully run against a system, the Master Control Record for that system is updated with the date/time of the unload. During an incremental unload, the date/time stamp on each element is compared with the date/time stamp on the Master Control Record for the system to which the element belongs. Only those elements with a date/time stamp that is more recent than the date/time stamp in the Master Control File system record will be selected during incremental unload processing.

Package Unloads

All packages contained in the package data set defined in the C1DEFLTS table will be unloaded. Optionally, individual packages may be selected for unloading using the ID parameter.

Historic versions of promotion packages are unloaded along with the current version. In the unload SCL, you cannot specify the package ID of individual historic versions of a promotion package. This ensures the integrity of the entire package. Always specify the fully qualified or mask package ID of the current version you want to unload. The unload utility unloads the records for all versions of that promotion package.

Validation During Unload

Minimal validation automatically occurs during Unload processing. This ensures that any element which is to be unloaded meets certain integrity criteria prior to being unloaded. No element found to be corrupt will be unloaded. Rather, an appropriate error message is issued indicating the exact nature of the problem. The same processing will occur if the Validate function is run independently.

If an error is detected during Unload processing, it is important to determine the cause of the error and to correct the problem. In most cases (a physical problem) the Reload facility should allow you to accomplish this task by going back to the last good unload point.

For all corrective action, see the *Messages and Codes Reference Guide*. For certain logical problems it may be necessary to contact CA Endevor SCM Customer Support.

If a validation error occurs during Unload processing the following occurs:

- If the validation problem affects the element itself, Unload will:
  - Not unload the element.
  - Write an error message indicating the nature of the problem.
  - Write an SCL DELETE statement to the C1SCL1 data set. The SCL contained in the C1SCL1 data set can then be used in a subsequent recovery operation to delete the element prior to performing Reload processing.
- If the element itself is intact, but there is a problem in the associated ACM component list, Unload will:
  - Unload the element.
  - Write an error message indicating the nature of the problem.
  - Not unload the component list for the element.
  - Write an SCL DELETE statement with the ONLY COMPONENT clause to the C1SCL1 data set. The SCL contained in the C1SCL1 data set can then be used in a subsequent recovery operation to delete only the element's component list prior to a GENERATE action or Reload processing.

Packages

No validation takes place during a package unload. The package data set is considered to be intact if each package ID can be read successfully.

Unload Recommendations

Unload is designed to easily capture all related parts of the CA Endevor SCM structure in a unified manner. This allows for timely physical recovery in the event of a device failure or site disaster.

Performing a needs analysis for your site is the first step in making optimal use of this utility. The criteria you should consider include the following:

- Number of environments/systems and related base and delta libraries at your site.
- Location of files in relation to DASD layout.
- Current schedule of in-house DASD backup and recovery procedures.
- Volatility and number of changes.

Use this information to determine how best to utilize and/or schedule Unload processing.

Locking During Unload Processing

Unload processing maintains a share lock (enqueue) at the environment level, while keeping an enqueue exclusive at the system level. This allows several unloads to run concurrently against different systems in the same environment.

Unload processing will not begin until the lock can be set (after all activity against the system has ceased). No CA Endevor SCM activity can resume until the unload for that system has been completed. The reload function still sets an enqueue exclusive at the environment level, thus it cannot be run while an unload is being performed.

Locking Example 1

A site has only one environment, with all systems pointing to one set of base and delta libraries. A low number of changes are made and there is little concern if a physical problem causes an outage during recovery.

In this situation it may be simpler to back-up all files on a daily basis using the appropriate in-house methods. In the event of a physical problem, CA Endevor SCM can be completely restored using the previous evening's backup files.

Locking Example 2

A site has a large number of environments/systems, each with their own base/delta libraries. Any outage will affect a large number of people in many areas of the company.

In this situation both full (FULL) and incremental (INC) unloads should be run. In most cases a full unload job should be scheduled on a weekly basis, and an incremental unload run each night. In the event of a physical failure it will be important to recognize which unload file(s) pertain to each system. Therefore appropriate naming conventions for the unload files should be adopted.

Note: It is important to align scheduling of Unload processing with in-house backup utilities; to determine what unload files need to be applied when a DASD volume is restored. In general, Unload processing should be run immediately following each backup utility run. This minimizes the number of "orphan" members that occur during a reload.

Sample Unload Control Cards

To specify a full unload from environment Test, code the following statement:

```
UNLOAD FULL FROM ENV TEST SYS * TO DDN UNLOD01.
```

In this example, CA Endevor SCM unloads all systems within this environment to an output data set with DDname UNLOD01. This data set must be coded in the Unload JCL.

To specify a package unload, code the following statement:

```
UNLOAD PACKAGE TO DDN UNLODPKG.
```

CA Endevor SCM unloads the entire package data set for this site to an output data set with ddname UNLODPKG. This data set must be coded in the Unload JCL. The JCL which unloads the package can be found in member BC1JUNLD, in the JCL library iprfx.iqual.CSIQJCL.

C1BM5000

The Unload/Reload/Validate Utility program.

CONLIB

Data set name of the installation CA Endevor SCM CONLIB load library.

BSTIPT01

Required DDname for Unload/Reload/Validate statements.

UNLODNN

This is the DDname for the data set to which the Unload information will be written. It must be allocated with a minimum LRECL=4200, a block size 4 bytes greater than 4200, and a RECFM=VB.

Note: The LRECL should be at least 84 bytes larger than the largest LREC you are unloading.

This data set should be set up as a Generation Data Group (GDG) with the proper number of levels based on unload frequency.

Note: We recommend that you adopt a naming convention based on the type of Unload being performed (FULL or INC) and the environment system to which it applies. For example:

FULL

BST.TEST.FINANCE.WEEKLY

INC

BST.TEST.FINANCE.DAILY

SPACE=(TRK, (NN,NN))

Required disk space for the unload data set (omit if tape). To estimate an approximate space allocation for this data set, add the following:

- 1105 bytes for each environment definition record on the Master File (1021-byte record length plus 84-byte prefix; the system unloads one environment record for each system, subsystem, type, type sequence, data set, and element master record), plus
- The current space taken up by your base and delta libraries (these libraries will continue to be written in compressed format), plus
- An 84-byte prefix for each base and delta element to be unloaded.

C1SCL1

Required DDname for the data set to which SCL DELETE statements will be written.

The Reload Function

The Reload function allows you to recover a CA Endevor SCM VSAM control file (Master Control File or package data set) or a base/delta data set that was lost as the result of a physical device failure or site disaster. Reload restores data from data sets created by the Unload process. The TRANSFER action may be used to transfer elements from an Unload file.

Note: For more information about the TRANSFER action see the *SCL Reference Guide*.

Reload processing occurs in two phases. Phase one reloads Master Control File environmental definitions (system, subsystem, type, type sequence, data set, and so on) in order to re-establish this information. Phase two reloads elements (element master record, base/delta/component list).

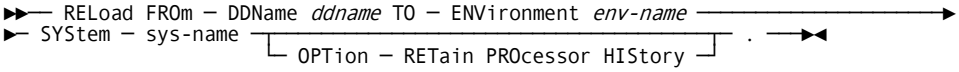
The Reload function, when used in combination with the batch Restore (SCL) action it can also be a powerful tool for recovering most logical and physical problems.

To make the best use of the Reload function, keep in mind the following:

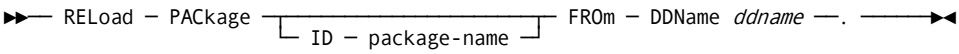
- Because no active journaling takes place, Reload processing cannot perform “point-of-failure” recoveries. It is intended to be used in conjunction with proper file/DASD layouts to reduce the impact, across all systems, of a base/delta failure, and to insure proper inventory synchronization if other outages occur.
- Reload processing replaces CA Endevor SCM data set or library contents with unload file contents when:
  - The date/time stamp of a record on the unload file is more recent than the one currently in CA Endevor SCM.
  - An element currently in CA Endevor SCM does not meet the Validation criteria (the record in CA Endevor SCM has a more recent date/time stamp but the Validation process finds a missing base member).
- This utility is not designed to be used to simply back off or back out a member in a data set (For more information, see Option 7: Backing Out/Backing in Package Outputs in the chapter “ Processing Packages in Foreground” in the *Packages*
- Reload can accept a concatenated input stream of unload files. Because of the date/time checking that occurs, only the latest data is reloaded. However, it is highly recommended that files be ordered from the oldest to the newest. This ensures that the logical cleanup of elements occurs. Failure to do this may result in elements ending up at both stages, out of logical order.
- The CA Endevor SCM RESTORE action can use an unload file as input. This can be used to perform logical recovery by first deleting the element, then restoring from the unload file.
- The Unload/Reload reports (CONRPT50-55) should be used to determine the status of an unload file. Use the reporting facility whenever you need to itemize the activity contained on one of these files.

Reload Control Card

The parameters for using Reload are as follows:



The parameters for using Reload for a package data set or one or more specific IDs are as follows:



The parameters for the Reload control card are described next:

RELOAD

Specifies the Reload function of program C1BM5000.

FROM DDNAME

Identifies the DDname to which the input Reload data set name will be assigned. Both environment and package processing require a DDname.

TO

Allows you to specify the to location for a combination of environment(s) and system(s) to be reloaded. The qualifying statements are:

- ENVIRONMENT-Identifies the environment(s) into which information is to be reloaded. A name mask may be used.
- SYSTEM-Identifies the system(s) into which information is to be reloaded. A name mask may be used.

RETAIN PROCESSOR HISTORY

Retains the Master Control File last processor information, indicating the date and time when the processor was last executed. If you are sure that the status of the processor output members matches the Master Control File history, you can use this option to retain the last processor information when executing the Reload utility. Do not use this option if the status of the processor output members is not known to be correct.

PACKAGE

Required when reloading a package data set or one or more packages. You may also select a package ID to identify the name of a specific package to be reloaded. You can use a name mask. If you do not specify a package ID, all packages found in the unload data set will be reloaded to the package data set defined in the C1DEFLTS table.

For promotion packages, you cannot specify the package ID of individual historic versions. You can specify the fully qualified name or mask package ID of the current version. This reloads each promotion package version that is missing or that needs to be reloaded due to date validation inconsistencies.

Reloading Master Control File Information

Reload determines the status of the VSAM Master Control File (MCF) and performs either a partial or complete reload by comparing each environmental record (excluding element records) on the unload file with the existing MCF.

If the VSAM Master Control File for the environment being reloaded is empty (new file allocated), Reload processing rebuilds the file using the contents of the unload data set(s).

If the Master Control File contains the environmental record, Reload processing does the following:

- If the system information on the unload file is more recent than the existing system information, the existing information is replaced.
- If the subsystem information on the unload file is more recent than the existing subsystem information, the existing information is replaced.
- If the type information on the unload file is more recent than the existing type information, the existing information is replaced. This is done as follows:
  - Information for existing types is not changed.
  - Type sequence types added from the unload file are sequenced after the existing
- If the approver group/relation information on the unload file is more recent than the existing approver group/relation information, the existing information is replaced.
- No existing data set information is replaced. Rather, Reload adds any data set information that is not found in the current Master Control File.

Reloading Element Information

After updating and/or rebuilding the Master Control File, elements and component lists will be reloaded based on the date/time stamps of existing elements and component lists.

- If the element or component list in the unload file is more recent than the existing MCF element or component list, Reload processing will:
  - Replace the existing element master record with the contents of the unload file.
  - Replace the existing base and delta members for the element and/or component list with the contents of the unload file.
  - Write an SCL GENERATE action request for that element to the C1SCL1 data set.
- If the element or component list in the unload file is not more recent than the existing MCF element or component list, Reload will perform validation processing on the element/component list. For a description of this validation process, see Unload Function in this chapter.
- If the element fails validation, Reload will:
  - Replace the existing element master record with the contents of the unload file.
  - Replace the existing base and delta members for the element and/or component list with the contents of the unload file.
  - Write an SCL GENERATE action request for that element to the C1SCL1 data set.
- If the element passes validation the element is not reloaded.
- If a Stage 2 element is reloaded and the element exists at Stage 1, reload processing will delete the Stage 1 element if the last action date for the Stage 2 element is more recent than the last action date for the Stage 1 element.

Reload and Packages

Reload will compare the date/time stamps of packages from the unload file with the date/time stamps of existing packages in the package data set.

- If the date/time stamp of the package from the unload file is the more recent than the one contained in the package data set, Reload will replace the existing package.
- If the date/time stamp of the existing package is more recent than the one contained in the unload file, Reload will not replace the package.

For promotion packages, the reload utility loads each of the versions that are missing or that need to be reloaded due to date validation inconsistencies.

For example, assume the unload utility was executed against the package file and the package ID of the current version is PKG1 and there are two historic version: EXAOBHX1 and EXAJHBX2. Assume the package records for EXAOBHX1 were deleted and the PKG1 and EXAJHBX2 package records remain intact. When you issue the reload for PKG1, the reload utility determines that only package EXAOBHX1 must be reloaded and reloads it.

Locking During Reload Processing

Reload processing will maintain an exclusive lock (enqueue) at the environment level. Reload processing will not begin until the lock can be set (after all activity against the environment has ceased). No CA Endevor SCM activity can resume until the reload for that environment has been completed, nor can reload processing run concurrently with UNLOAD processing.

Example 1. Base/Delta Recovery

Problem: A DASD volume has a hardware failure Tuesday at 10:00 a.m. This pack contained the only base and/or delta libraries defined for all systems in this environment. The volume was backed-up using an in-house backup utility, and unloaded using an incremental unload the evening before. The previous Sunday a full unload was performed.

Solution: Restore the DASD volume from the latest in-house backup file. Once this has been done, determine to what point the volume was restored (date/time) in relation to the unload files that you have available.

- If no changes have been made to CA Endevor SCM data since the last backup, the files should be considered recovered (in sync), and CA Endevor SCM processing may be continued.
- If only environmental changes have been made to CA Endevor SCM since the last backup (no element updates were made), the files should be considered recovered (in sync), and CA Endevor SCM processing may be continued.

- If element modifications have been made since the last backup, an out-of-sync condition will exist between the base and/or delta libraries and the Master Control File (MCF), requiring the system(s) to be reloaded. To do this:
  - Itemize the system(s) affected by the element modifications.
  - Set up a Reload job using as input the last full unload file, and all incremental files that are available up to the point of the failure.
  - Run Reload processing for all the affected system(s) using the concatenated input of unload files.
  - Use the SCL file (C1SCL1) produced during Reload to run the GENERATE action against the reloaded elements. This will synchronize the outputs.

Once the reload has been completed, Validate processing must be run for each system that was reloaded. This is necessary to identify "orphan" members (members that had been added since the recovery point). To do this:

- Run Validate processing for all the affected system(s).
- Use the SCL file (C1SCL1) produced by the Validate process to delete each element that failed validation and complete the synchronization process.
- **Note:** The deleted elements can be manually recovered from a source output library member or an external copy of the source. If a source output library exists, ensure that the members to be recovered are first saved from this library before delete processing is performed. All other changes that have been made since that time will have been effectively "rolled back."

Example 2: VSAM Master Control File Recovery

Problem: A DASD volume has a hardware failure Tuesday at 10:00 a.m. This pack contained the VSAM Master Control file for one CA Endevor SCM environment. The volume was backed-up using an in-house backup utility, and unloaded using an incremental unload the evening before. The previous Sunday a full unload was performed.

Solution: Restore the DASD volume from the latest in-house backup file. Once this has been done, determine to what point the volume was restored (date/time) in relation to the unload files that you have available.

- If no changes have been made to CA Endevor SCM since the last backup, the files should be considered recovered (in sync), and CA Endevor SCM processing may be continued.
- If only environmental changes have been made to CA Endevor SCM since this last backup (no element updates were made) the files should be considered recovered (in sync). Any environmental changes made since the recovery point will have been lost and must be manually recovered. Once this has been done CA Endevor SCM processing may be continued.

- If element modifications have been made since the last backup, an out-of-sync condition will exist between the Master Control File (MCF) and the base and/or delta library, requiring the system(s) to be reloaded. To do this:
  - Itemize the system(s) affected by the element modifications.
  - Set up a Reload job using as input the last full unload file, and all incremental files that are available up to the point of the failure.
  - Run Reload processing for all the affected system(s) using the concatenated input of unload files.
  - Use the SCL file (C1SCL1) produced during Reload to run the Generate action against the reloaded elements. This will synchronize the associated outputs.

Note: Elements added since the recovery point can be manually recovered from the contents of a source output library member or an external copy of the source. All other changes that have been made since that time will have been effectively "rolled back."

Note: If the reloaded SYSTEM requires CCID and/or COMMENT to be specified for actions against its elements, you must insert a SET OPTIONS statement in the generated SCL file (C1SCL1).

Example 3: Package Data Set Recovery

Problem: A DASD volume has a hardware failure on Tuesday at 10:00 A.M. This pack contained the CA Endevor SCM package data set. The volume was backed up using an in-house backup utility the evening before.

Solution: Restore the DASD volume from the latest in-house backup files. Once this has been done, determine to what point the volume was restored (date and time) in relation to any package unload files you may have available.

- If no changes have been made to the package data set since the last backup, the data set should be considered recovered (in sync), and CA Endevor SCM processing may be continued.
- If any package modifications have been made since the last backup, they will be lost. If there is an unload file available that is more recent than the backup from which the data set was restored you may elect to reload the package data set. To do this:
  - Set up a Reload job, using as input any unload file that is more recent than the backup file used to restore the data set.
  - Run Reload processing for all the affected package(s) or selected package IDs.

Sample Reload Control Cards

The following request specifies that environment TEST and all systems in this environment are to be reloaded. UNLOD01 is the DDname assigned to the input data set that contains the data to be reloaded. This data set must be coded in the Reload JCL.

```
RELOAD FROM DDN UNLOD01 TO ENV TEST SYS *.
```

UNLODPKG is the DDname assigned to the input data set that contains the data to be reloaded. This data set must be coded in the Reload JCL. The JCL which unloads the package can be found in member BC1JRELD, supplied in your iprfx.iqual.CSIQJCL library.

C1BM5000

The Unload/Reload/Validate Utility program.

CONLIB

Data set name of the installation CA Endevor SCM CONLIB load library.

BSTIPT01

Required DDname for Unload/Reload/Validate statements.

UNLODNN

The target DDname of the unload data set name specified for this Reload operation.

C1SCL1

Required DDname for the data set to which the SCL GENERATE statements produced by the Reload operation will be written.

C1RLDWK*

Pre-allocates work files so that site does not have to rely on CA Endevor SCM for temporary allocation.

The Validate Function

The Validate function allows you to ensure the integrity of one or more existing CA Endevor SCM environments and/or systems and their related elements and components.

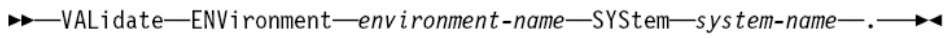
The Validate function performs a series of checks against the contents of the VSAM Master Control File(s), and the related base and delta libraries associated with the environments and systems specified on the job request. These are the same checks performed as part of Unload processing, allowing this function to operate in a standalone mode.

Validate may be run for an entire environment, or for selected systems within an environment. There is no validation processing currently available for the package data set.

The Validate function should be run any time there is a question about the integrity of an environment/system, and in conjunction with a reload operation. Note that an unload does not do the same checking against the files that a validate does.

Validate Control Card

The parameters for the Validate control card are as follows:



The parameters for the Validate control card are described next:

VALIDATE

Specifies the Validate function of program C1BM5000. You must further qualify the Validate request with one of the following:

- ENVIRONMENT-Identifies the environment(s) for which information is to be validated. A name mask may be used.
- SYSTEM-Identifies the system(s) from which information is to be validated. A name mask may be used.

Any number of Validate statements may be coded for a single run.

Validate Processing

Validate processing performs a series of checks against the contents of the VSAM Master Control File(s), and the related base and delta libraries. These checks include:

- MCF to base to delta relationship verification.
- Footprint correlation.
- Physical base/delta member counts.
- Insert/delete counts.
- Delta level verification.

If an error is detected it is important to determine the cause and to correct the problem. In most cases (a physical problem) the Reload facility should allow you to accomplish this task by going back to the last good unload point.

For all corrective actions, see the *Errors and Messages Guide*. For certain problems it may be necessary to contact CA Endevor SCM Customer Support.

Validate checks Master Control File and element information in the following order:

1. Stage 1 internal environmental definitions.
2. Stage 2 internal environmental definitions.
3. Stage 1 elements.
4. Stage 2 elements.

If an error is encountered during Validate processing the following will occur:

- If the validation problem affects the element itself, Validate will write:
  - An error message indicating the nature of the problem.
  - An SCL DELETE statement to the C1SCL1 data set. The SCL contained in the C1SCL1 data set can then be used in a subsequent recovery operation to delete the element prior to performing Reload processing.
- If the element itself is intact, but there is a problem in the associated ACM component list, Validate will write:
  - An error message indicating the nature of the problem.
  - An SCL DELETE statement with the ONLY COMPONENT clause to the C1SCL1 data set.

The SCL contained in the C1SCL1 data set can then be used in a subsequent recovery operation to delete only the element's component list prior to a GENERATE action or Reload processing.

Sample Validate Control Card

The following request specifies that environment Test and all systems in this environment are to be validated:

```
VALIDATE ENV TEST SYS *.
```

The JCL which validates the package can be found in member BC1JVALD, supplied in your iprfx.iqual.CSIQJCL library.

C1BM5000

The Unload/Reload/Validate Utility program.

CONLIB

Data set name of the installation CA Endevor SCM CONLIB load library.

BSTIPT01

Required DDname for Unload/Reload/Validate statements.

C1SCL1

Required DDname for the data set to which the SCL DELETE statements produced during Validate processing will be written.

Chapter 13: Using the Comma Separated Value (CSV) Utility

This section contains the following topics:

The CSV Utility

The Comma Separated Value (CSV) utility lets you extract Master Control File (MCF) and package file information and write it to a CSV formatted file. The CSV file can be downloaded and imported into various software products to create reports. You can extract the following types of information:

- List approver group
- List approver group junction (approver relation)
- List components used by element
- List data set
- List destination
- List element
- List elements using component
- List environment
- List package ID
- List package action summary
- List package approver group
- List package SCL
- List package ship
- List processor group
- List SMF data
- List stage
- List subsystem
- List system
- List type

The CSV utility runs as a stand-alone program under NDVRC1. You build or modify the SCL statements that determine the CSV output. The utility analyzes the contents of the SCL statements passed by means of the CSVIPT01 DD statement. Then it calls API functions to obtain the data. Finally, it reformats the API response block data into the CSV format and returns it to the API where the data is written to the CSV list file.

CSV Input

SCL is used as input to the CSV Utility to request functions. The SCL follows the same conventions as batch requests.

Note: For a complete list of the syntax rules, see the *SCL Reference Guide*.

CSV supports name masking for some values. For details, see the syntax description for each function. The placeholder character (%) and partial and full wildcarding are supported in many of the FROM parameters. You can use either of the following methods to code full wildcarding in the FROM parameters:

- Code the keyword followed by an asterisk (*), or
- Omit the keyword and associated value from the SCL statement

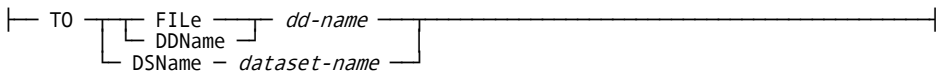
Comments can be added to CSV syntax by placing an asterisk in column one of an SCL statement.

Common TO Clause

The TO clause can be specified in the syntax for each CSV function. The purpose of this clause is to allow you to override the name of the file where the output will be returned. If a TO clause is not specified, the default DDNAME used is APIEXTR. If you run a CSV job containing more than one action and do not specify different list files, only the results of the last action will appear in the list file. If you intend to write the results of more than one request to the same file, you can do this by specifying the MOD sub-parameter on the DISP parameter in the JCL.

If you override the list file name, make sure to add the additional DD statements to the JCL.

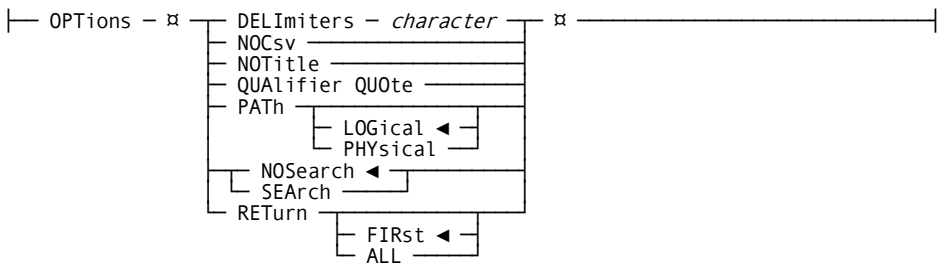
The syntax of the TO clause is shown next:



Common OPTIONS Keywords

The OPTIONS clause can be specified in the syntax for each CSV function. The OPTIONS clause allows you to specify OPTIONS parameters to override default values and to tailor the specifications of the function.

The syntax of the OPTIONS clause is shown next:



OPTIONS Parameters Common to All CSV Functions

Four parameters are common to all functions. You can use any of the following parameters in any function statement:

DELImiters *character*

Specifies the one-character value that is used to separate values. The default value is a comma (,).

NOCsv

Formats the CSV utility output in record format, instead of CSV format, that you can read into a program. NOCSV enables you to avoid having to write an API program to extract the output in record format. With NOCSV, the output is formatted the same as the API response structure. No title line appears in the file, no delimiters between values, no quotes surrounding values, and so on. Also the API response header appears at the beginning of each record. For a description of the API response blocks, see the *API Guide*.

When the NOCSV option is specified, the BACKOUT field does not appear in the output. When extracting element information using the NOCSV option, ensure that you know whether the element is part of a backed-out package. To determine if the element is part of a backed-out package, examine the program that reads the data to see if the source and target package names, dates, and times are the same. If the source and target are identical, the package is not backed out. If any of the values are different, the element is part of a backed-out package.

Important: If you upgrade to a new release that changes the output of a CSV function and you have existing CSV jobs that use the NOCSV option, you may need to change your existing programs that read in the CSV output file due to record layout changes.

For API functions that change with a new release, the output from the functions is upward-compatible. Therefore, if you have existing API programs and you do not recompile them after installing the new release, the response structures produced will be identical to the output produced by the previous release. There is no need to reassemble and relink your programs, unless you plan to use the new release enhancements for that function.

This is not the case for CSV functions. CSV output always matches the format of the latest release, because CSV functions call the current Endevor API to produce the output, which is always in sync with the latest release. Therefore, to accommodate changes in the CSV output, you may need to change your existing programs that read in the CSV file.

Changes to API or CSV functions are listed in the *Release Notes*.

NOTitle

By default, a title line is written as the first record in the list file. To suppress the printing of this record, specify the NOTITLE parameter.

QUAlifier QUOte

By default, double quotes are placed around each value in the list file. To change this character to a quote, specify the QUALIFIER keyword followed by the keyword QUOTE.

OPTIONS Parameters Common to Some CSV Functions

You can use the following three parameters for only some of the functions. These parameters are shown in the syntax section for each of the functions for which they are valid.

PATh LOGical/PHYsical

Mapping path. Determines if CA Endevor SCM should use the physical or logical mapping path as defined in the C1DEFLTS table. The default is logical.

NOSearch/SEArch

Mapping argument. Search across mapping locations or only search at the location specified. The default is nosearch.

RETurn FIRst/ALL

Return only the first or all records that satisfies the request. This parameter is used in conjunction with an object that exists at more than one location. The default is first.

The EOF or EOJ Statement

The EOF or EOJ statement instructs CA Endevor SCM to stop parsing the SCL syntax at a particular point. Using this statement eliminates the need to manually delete any statements you do not want CA Endevor SCM to process. If no EOF or EOJ statement is coded, CA Endevor SCM generates one after the last SCL statement.

CSV Output

Listed next is a sample of how the output appears in the CSV list file. A title record appears first, followed by each detailed data record. In the sample, the "..." that appears at the end of each line represents additional columns.

```
"SITE ID","ENV NAME","SYS NAME","SBS NAME","STG NAME",...
"0","I40","NDVRMVS","BASE","I40STG1",...
"0","Q40","NDVRMVS","BASE","Q40STG2",...
"0","P40","NDVRB40","BASE","P40STG2",...
```

The CSV output layouts have the following attributes:

- The first record returned in the output file will always be a title record, unless the NOTITLE option is present in the request.
- Values are separated by a delimiter character. The default separation character is a comma. The DELIMITER option can be used to override the default character.
- Each value is placed in double quotes. The QUOTE option can be used to override this default to change the characters to single quotes.
- Numeric values are leading zeros suppressed. Character data containing leading zeros is unaffected.
- The date is always returned in the yyyy:mm:dd format.
- Time format is always returned in hh:mm:ss:tt format unless otherwise stated. For package functions where only the hour and minutes are available, hh:mm is returned.

CSV Message Reports

CSV related messages are written to one or two of the following three message files:

- CSVMSGS1
- C1MSGSA
- C1MSGS1

If the CSVMSGS1 file is coded, the CA Endevor SCM Syntax Request Report is written to the CSVMSGS1 file. If the CSVMSGS1 file is not present, the CA Endevor SCM Syntax Request Report is written to the C1MSGS1 file. The CSV utility uses CSVMSGS1 if both files are present, and fails if neither of the files is present.

If DDNAME C1MSGSA is coded, the API Execution Report is written to this file. If it is not coded, the CSV utility writes the API Execution Report to the same file as the Syntax Request Report.

The CA Endevor SCM Syntax Request Report echoes the SCL statements that were coded and lists any syntax errors that were detected.

The CA Endevor SCM API Execution Report contains informational, warning, and error messages detected while the request was being processed. All field edit and processing messages are written to this report. Return and reason code information, and selection counts are displayed in this report.

Note: For detailed message information, see the *API Guide* and the *Messages and Codes Reference Guide*.

Title Line Abbreviations

Standard literal abbreviations are used in the title line written as the first record in the CSV output file. The following lists the standard abbreviations used and the full word each is associated with:

#- Number

(F)- From location

(ISO)- International Standards Organization

(S)- Source location

(T)- To location

ACT- Action

APPR- Approver /Approval

BYP- Bypass

CHG- Change

CMPL- Complementary

CMPNT- Component

CNT- Count

CONSOL- Consolidate

CORR- Correlation

CUR- Current

DEL- Delete

DEST- Package shipment destination

DS- Data set

DSN- Data set name

ELM- Element

ENV- Environment

ERR- Error

EXEC- Execute or Execution

EXPND- Expand

EXT- Extension

FG- Foreground

FLG- Flag

FWD- Forward

GEN- Generate

GRP- Group

HDR- Header

HIST- History

IGN- Ignore

INCL- Include

(ISO)- International Standards Organization

LANG- Language

LIB- Library

LL- Level

LL.VV- Level.Version

LNG- Length

LOC- Location

MBR- Member

MTH- Month

O/P- Output

OPT- Option

OVRD- Override

PKG- Package

PREV- Previous

PROC- Processor

RC- Return Code

RCD- Record

RCV- Receive

REG- Registration

REGR- Regression

REL- Release ID

REQ- Required

RETR- Retrieve action

REV- Reverse

RMT- Remote

SBS- Subsystem

SEV- Severity

SRC- Source

STG- Stage

STMTS- Statements

SYM- Symbol/Symbolic

SYS- System

TBL- Table

TERM- Termination

TRANS- Transmission

TRNSFR- Transfer

UPDT- Update

USR- User

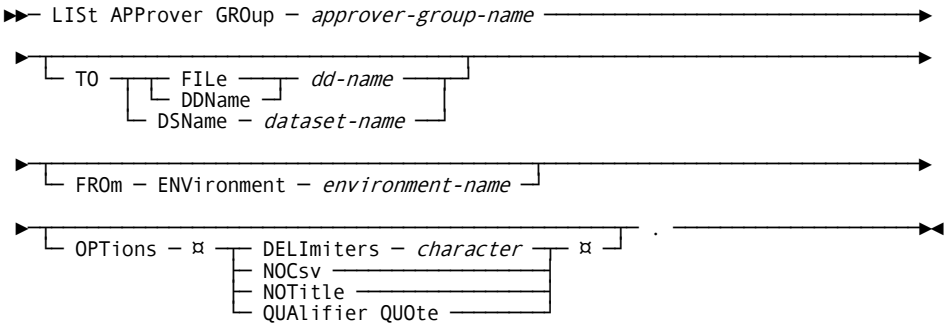
VIO- Violation

VV- Version

The List Approver Group Function

The list approver group CSV function allows you to extract approver group information from the MCF that satisfies the criteria you request.

The syntax of the list approver group CSV function is shown next:



Wildcard and/or placeholder characters are permitted in the environment and approver group name fields. Approver group information is extracted individually from each environment that meets its selection criteria. Also, the FROm ENVironment clause is optional, and if omitted, the environment name is wild.

Example

```
LIST APPROVER GROUP * TO FILE CSVFILE.
```

Result for this example: If 2 environments (ENVA and ENVB) are defined in the C1DEFLTS table, all approver groups defined in ENVA will be extracted first, then all from the second environment, ENVB, will follow.

Extracted Approver Group Data

The output file contains the following types of information:

SITE ID

Site ID

ENV NAME

Environment name

STG NAME

Stage name

STG ID

Stage ID

STG #

Stage number (1/2)

APPR GRP NAME

Approver group name

RCD UPDT CNT

Record update count

UPDT DATE

Last Update date YYYY/MM/DD

UPDT TIME

Last Update time HH:MM:SS:TH

UPDT USRID

User ID associated with last update

REL ID

Record created release ID

QUORUM CNT

Quorum count

TITLE

Approver group title

APPR USRID

Approver user ID. A maximum of 16 approvers can exist for each approver group. The output file will contain one APPR USRID field for each approver. Each APPR USRID field is followed by a APPR REQ.

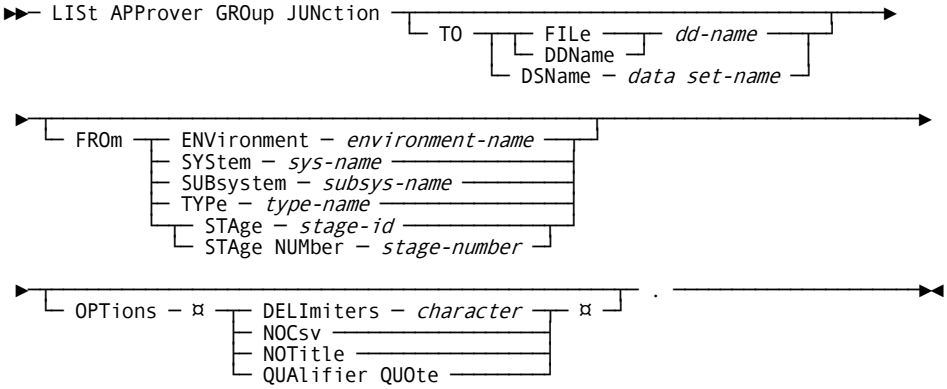
APPR REQ

Required approver flag (Y/N). The output file will contain one APPR REQ field for each APPR USRID. The APPR USRID field and APPR REQ field appear in pairs.

The List Approver Group Junction Examples

The list approver group junction CSV function allows you to extract approver group relation information from the MCF that satisfies the criteria you specify.

The syntax of the list approver group junction CSV function is shown next:



FROM ENVIRONMENT environment-name SYSTEM system-name SUBSYSTEM subsystem-name TYPE type-name STAGE ID stage-id | STAGE NUMBER stage-no

Specifies the inventory location for which approver relation information is being extracted.

ENVIRONMENT

Specify an explicit or name-masked value. A name-masked value can include a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. You can also fully wildcard the value by leaving the value blank (for example, ENV " "), or by leaving out the ENV keyword and value.

SYSTEM, SUBSYSTEM, and TYPE

Specify these values explicitly (for example, ABC) or specify a wild value by leaving the clause out or leaving the value blank (for example, SYSTEM ‘ ‘). All characters pass the parser check, but wildcard (*) or placeholder (&) characters are not expanded to find matching definitions. You can specify "*" or "&" but these are treated as text values.

STAGE ID stage-id | STAGE NUMBER stage-no

Specify the stage using one of the following stage keywords. If you do not specify either of the keywords and their values, wildcarding is enforced.

STAGE ID

Enter a single alphanumeric stage identifier. Do not use if the Environment is wild. However, if the Environment is not wild, you can specify STAGE ID "*" but the * is treated as a text value instead of a wildcard.

STAGE NUMBER

Enter *, %, 1, 2, or leave the value blank. However, only a blank is treated as wild. The * and % values are treated as text values.

List Approver Group Examples

The following clauses are equivalent.

```
LIST APPROVER GROUP JUNCTION * TO FILE CSVFILE
    FROM ENV ENV*.

LIST APPROVER GROUP JUNCTION * TO FILE CSVFILE
    FROM ENV ENV* SYS ' ' SUB ' ' TYP ' ' STA ' '.
```

Results for these examples: If 2 environments (ENVA and ENVB) are defined in the C1DEFLTS table, all approver group junctions defined in ENVA will be extracted first, then all from the second environment, ENVB, will follow.

Extracted Approver Group Junction Data

The output file contains the following types of information:

SITE ID

Site ID

ENV NAME

Environment name

SYS NAME

System name

SBS NAME

Subsystem name

TYPE NAME

Type name

STG #

Stage number (1/2 or *)

RCD UPDT CNT

Record update count

UPDT DATE

Update date YYYY/MM/DD

UPDT TIME

Update time HH:MM:SS:TH

UPDT USRID

Update user ID

REL ID

Record created release ID

JUNCTION TYPE

Junction type; ST for standard or EM for Emergency

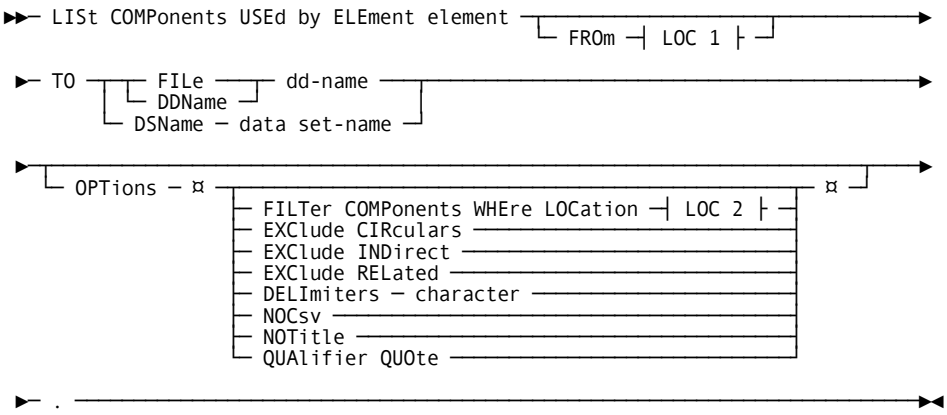
APPR GRP NAME

Approver group name

The List Components Used by Element Function

The list components used by element function lets you list the components used by the element you specify.

This action has the following format:



Parameters

This action uses the following parameters:

LISt USEd COMPonents FOR ELEment element *LOC*

Extracts a list of components that use the specified element. Optionally, you can limit the search by specifying the element's location.

OPTions

FILTer COMPonents WITh *LOC*

Filters the ACM data to return only those elements found at the specified location. This option eliminates all non-element output records (member, object, and comment) from the output. It eliminates all related elements also. This is true even if you wildcard all the location values.

EXClude CIRculars

Filters the ACM data to exclude components that have a circular relationship to the object of your search.

EXClude INDirect

Filters the ACM data to exclude indirectly related components.

EXClude RELated

Filters the ACM data to exclude related components.

LOC

The location fields further qualify the element query object. You can specify the location by the environment, system, subsystem, or type.

ENVironment *environment-name*

Specifies the one- to eight-character name of the environment in which you want to perform your query.

SYStem *system-name*

Specifies the one- to eight-character name of the system in which you want to perform your query.

SUBsystem *subsystem-name*

Specifies the one- to eight-character name of the subsystem in which you want to perform your query.

TYPe *type-name*

Specifies the type of the elements for which you are searching. The type name can be one to eight characters in length.

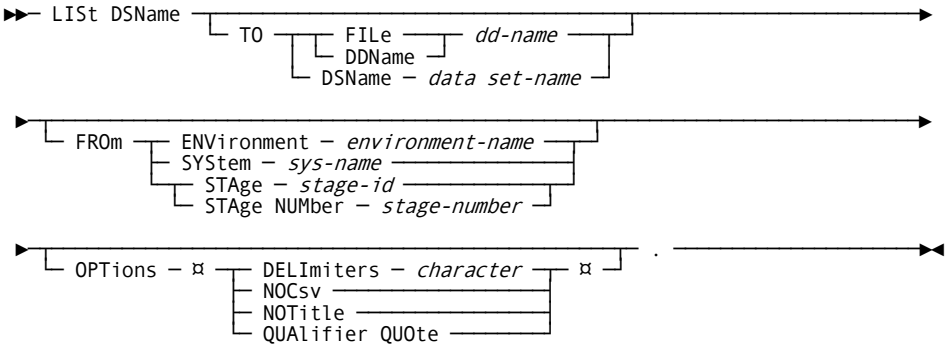
STAge NUMber *stage-number*

Specifies the number of the stage in which you want to perform your query.

The List Data Set Function

The list data set CSV function allows you to extract data set information from the MCF that satisfies the criteria you specify.

The syntax of the list data set CSV function is shown next:



Wildcard and/or placeholder characters are permitted in any of the environment, system, and stage location fields. For each location that meets its selection criteria, data set records are extracted based on the system name criteria. Location is defined as environment and stage.

The FROM clause is optional. All or part of the keyword specification on this clause can be omitted. On an omitted keyword, a wildcard will be assumed. Also, when the environment name is wild, the STAge NUMber clause cannot be used. You can either omit the stage clause or specify it through the STAge stage-id clause.

Example

```
LIST DSN TO FILE CSVFILE
FROM ENV * STA * SYS NDVRMVS.
```

Result for this example: If 2 environments (ENVA and ENVB) are defined in the C1DEFLTS table, data set records will be extracted in the following order:

1. all records from ENVA, stage 1 under system NDVRMVS
2. all records from ENVA, stage 2 under system NDVRMVS
3. all records from ENVB, stage 1 under system NDVRMVS
4. all records from ENVB, stage 2 under system NDVRMVS

Extracted Dataset Name Data

The output file contains the following types of information:

SITE ID

Site ID

ENV NAME

Environment name

SYS NAME

System name

STG NAME

Stage name

STG ID

Stage ID

STG #

Stage number (1/2 *)

DS RCD ID

Data set record ID

DS RCD TYPE

Data set type; PO, PV, LB, EL, VK, ??. If symbolics appear in the DS NAME field, the RCD TYPE value will be ?? for unknown.

DS NAME

Data set name

RCD UPDT CNT

Record update count

UPDT DATE

Update date YYYY/MM/DD

UPDT TIME

Update time HH:MM:SS:TH

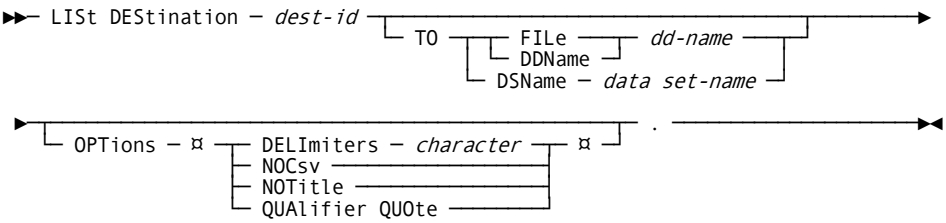
UPDT USRID

Update user ID

The List Destination Function

The list destination CSV function allows you to produce a list of destinations that packages can be shipped to.

The syntax of the list destination CSV function is shown next:



You can specify a wildcard character (*) as the last character in the destination field. You can also use one or more placeholder characters (%) in this field. Do not leave the destination field blank, because that results in a syntax parser error condition.

Extracted Destination Data

The output file contains the following types of information:

DEST ID

Destination id

DESCRIPTION

Destination description

TRANS CODE

Transmission method code Possible values are:

B

Bulk Data Transfer Program, Version 2

F

NetView File Transfer

L

Local (IEBCOPY)

M

Network Data Mover

N

NetView Distribution Manager

S

Bulk Data Transfer Program via NJE/NJI

X

XCOM

TRANS DESC

Transmission method description

TRANS NODE

Transmission node name

CMPL FILES

Complementary files flag (Y/N)

HOST DSN PREFIX

Host data set name prefix

HOST DSN DISP

Host data set disposition. Final disposition of data set. Possible values are DELETE or KEEP

HOST DSN UNIT

Host data set device unit

HOST DSN VOLSER

Host data set volume serial number

REMOTE DSN PREFIX

Remote data set name prefix

REMOTE DSN DISP

Remote data set disposition. Final disposition of data set. Possible values are DELETE or KEEP

REMOTE DSN UNIT

Remote data set device unit

REMOTE DSN VOLSER

Remote data set volume serial number

SITE ID

Site id

CREATE DATE

Record creation date YYYY/MM/DD

CREATE TIME

Record creation time HH:MM

CREATE USRID

User id associated with create

UPDT DATE

Last record update date YYYY/MM/DD

UPDT TIME

Last record update time HH:MM

UPDT USRID

User ID associated with last update

IPNAME

IPNAME specified on the destination definition for an XCOM transmission

IPPORT

IPPORT specified on the destination definition for an XCOM transmission

HOST USS PREFIX

Host path name prefix

DISP

Host path name disposition. Final disposition of the path. Possible values are DELETE or KEEP

REMOTE USS PREFIX

Remote path name prefix

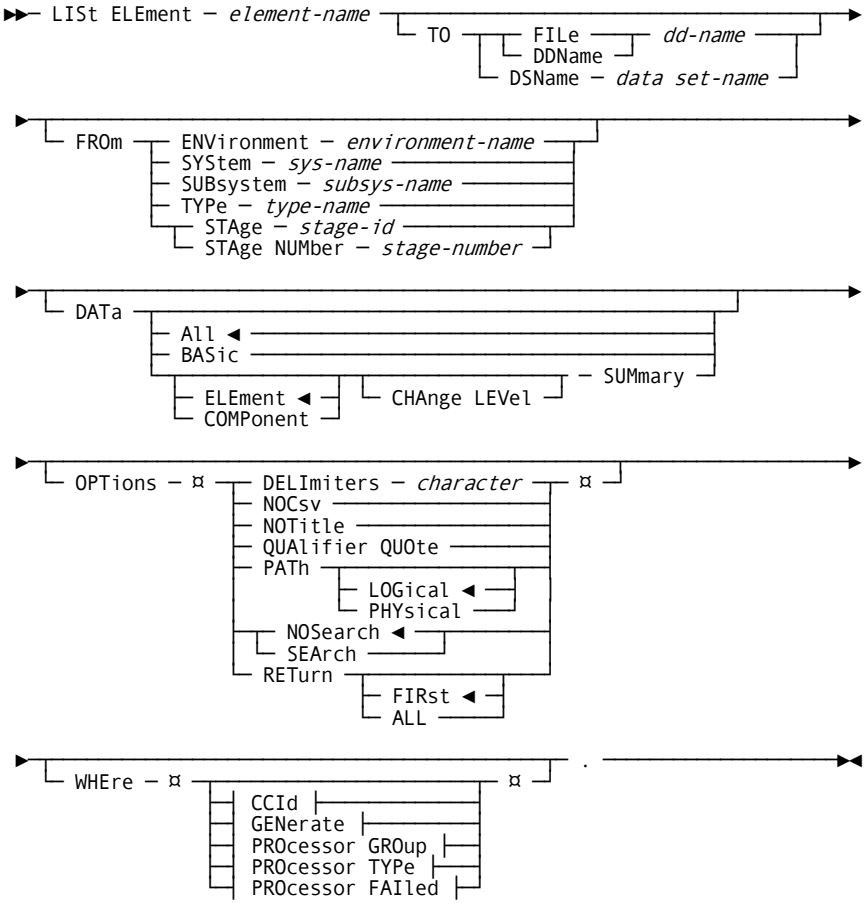
DISP

Remote path name disposition. Final disposition of the path. Possible values are DELETE or KEEP

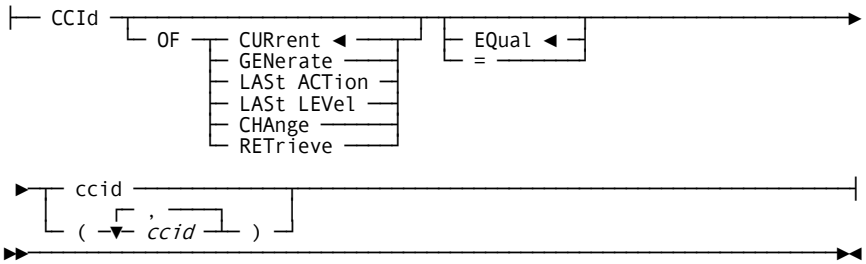
The List Element Function

The list element CSV function allows you to extract element information from the MCF that satisfies the criteria you specify.

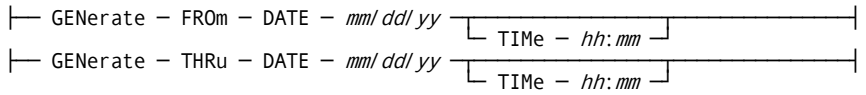
The syntax of the list element CSV function is shown next:



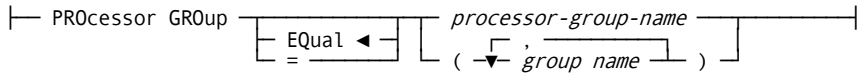
Expansion of CCID



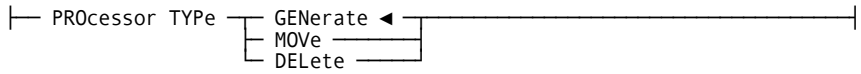
Expansion of GENERATE



Expansion of PROCESSOR GROUP



Expansion of PROCESSOR TYPE



Wildcard and/or placeholder characters are permitted in the environment, system, subsystem, type, element, and stage fields. On non-explicit environment name specifications, elements are extracted based on environment and stage locations, mapping between environment and stage locations is not supported.

Default mapping options on non-explicit environment name requests are: PATH PHYSICAL, NOSEARCH and RETURN ALL. The PATH LOGICAL, SEARCH and RETURN FIRST clauses are invalid on non-explicit environment name requests.

Default mapping options on explicit environment name requests are: PATH LOGICAL, NOSEARCH and RETURN FIRST.

The FROM clause is optional. All or part of the keyword specification on this clause can be omitted. On an omitted keyword, a wildcard is assumed. Also, when the environment name is wild, the STAge NUMber clause cannot be used. You can either omit the stage clause or specify it through the STAge stage-id clause.

The DATA clause is optional. If not specified, the default is DATA ALL. However, DATA ALL does not return change level summary data. To return element change level summary data, specify DATa SUMmary or DATa ELEment CHAnge LEVel SUMmary. To return component change level summary data, specify DATa COMPonent SUMmary or DATa COMPonent CHAnge LEVel SUMmary.

Only one WHERE CCID clause is allowed. If you attempt to specify more than one, a syntax error is generated.

If the WHERE CCID statement does not contain an OF parameter, the default OF CURRENT is assumed.

CURRENT CCID compares the CCID value(s) specified against the last action, generate, and last level CCIDs associated with an element. If any match is found, the record is selected.

One to eight CCID values can be specified on the WHERE CCID clause. To specify more than one value, place the values in ( ) and separate each value by a comma. For example: CCID OF GEN = (c1,c2,c3,c4,c5,c6,c7,c8).

The WHEre CCId OF CHAnge clause is only valid when the DATa SUMmary or DATa COMPonent SUMmary option is specified. This clause filters the results of the list data summary function based on the specified ccids.

Each change level for an element contains a CCID associated with that change. The CCID for each change level is compared against the ccids entered for the WHEre CCId OF CHAnge clause and only those change level summary records that match are returned by the list data summary function.

WHERE GEN clause: To select based on an exact match, code both the GEN FROM and GEN THRU clauses with identical date and time values. If you specify the FROM clause and no THRU clause, records starting at the FROM date and time and later are selected. If you specify the THRU clause and no FROM clause, records starting at the THRU date and time and earlier are selected.

One to eight processor-group-names can be specified on the where processor group parameter. To specify more than one value, place the values in ( ) and separate each value by a comma. For example: PRO GRO = (p1,p2,p3,p4,p5,p6,p7,p8).

Placeholder characters and a wildcard character are allowed in the CCID and PROCESSOR GROUP values.

List Element Example

```
LIST ELEMENT * TO FILE CSVFILE
   FROM ENV * STA * SYS * SUB * TYP *.
```

Result for this example: If 2 environments (ENVA and ENVB) are defined in the C1DEFLTS table where SYS1/SBS1 and SYS2/SBS2 are defined to both environment, element records will be extracted in the following order:

1. all element records from location ENVA, STAGE 1 under SYS1and SBS1
2. all element records from location ENVA, STAGE 1 under SYS2 and SBS2
3. all element records from location ENVA, STAGE 2 under SYS1 and SBS1
4. all element records from location ENVA, STAGE 2 under SYS2 and SBS2
5. all element records from location ENVB, STAGE 1 under SYS1 and SBS1
6. all element records from location ENVB, STAGE 1 under SYS2 and SBS2
7. all element records from location ENVB, STAGE 2 under SYS1 and SBS 1
8. all element records from location ENVB, STAGE 2 under SYS2 and SBS2

Extracted Element Data

The following information is contained in the output file. All the information is returned when the ALL option is specified on the DATA clause. Only the basic element information is returned when the BASIC option is specified. ALL is the default, if no DATA clause is specified.

When the option Element Change Level Summary or Component Change Level Summary is specified on the Data clause, only the fields specified in the Change Level Summary section are returned.

Basic Element Information

RCD TYPE

Type of response record; M-Master record; B-Basic Master record. Value determined by DATA clause parameter.

SITE ID

Site ID

ENV NAME

Environment name

SYS NAME

System name

SBS NAME

Subsystem name

ELM NAME

Element name

FULL ELM NAME

Full element name

TYPE NAME

Type name

STG NAME

Stage name

STG ID

Stage ID

STG #

Stage number (1/2)

STG SEQ #

Relative stage number (1/2/3,4…..)

PROC GRP NAME

Processor group name

UPDT DATE

Last physical record update date YYYY/MM/DD

UPDT TIME

Last physical record update time HH:MM:SS:TT

SIGNOUT ID

Signout user ID

ELM VV

Current element version number

ELM LL

Current element level number

CMPNT VV

Current component version number

CMPNT LL

Current component level number

Release ID Information

REL ID

Record created release ID

Last Action Information

LAST ACT MOD ELM

Last element modifying action

LAST ACT

Last action

ENDEVOR RC

CA Endevor SCM return code

LAST ACT DATE

Last action date YYYY/MM/DD

LAST ACT TIME

Last action time HH:MM:SS:TT

LAST ACT USRID

Last action userid

LAST ACT CCID

Last action CCID

LAST ACT COMMENT

Last action comment

Element Base Information

ELM BASE MBR

Element base member name

ELM BASE DATE

Element base member date YYYY/MM/DD

ELM BASE TIME

Element base member time HH:MM:SS:TT

ELM BASE # STMTS

Number of statements in base

BASE LL

Base level number

BASE COMPRESSED

Element base compressed flag (Y/N)

BASE USRID

User ID associated with the base level

BASE COMMENT

Comment associated with the base level

Element Delta Information

ELM DELTA MBR NAME

Element delta member name

ELM LAST LL DATE

Element delta member date YYYY/MM/DD

ELM LAST LL TIME

Element delta member time HH:MM:SS:TT

ELM LAST LL # STMTS

Number of statements in delta

ELM LAST LL USRID

User ID associated with the delta level

ELM LAST LL CCID

CCID associated with the delta level

ELM LAST LL COMMENT

Comment associated with the delta level

ELM LAST LL # INSERT

Number of inserts in last level

ELM LAST LL # DELETE

Number of deletes in last level

ELM DELTA FORMAT

Delta format (F/R/I)

Component Base Information

CMPNT MBR

Component base member name

CMPNT DATE

Component base dateYYYY/MM/DD

CMPNT TIME

Component base time HH:MM:SS:TT

CMPNT BASE # STMTS

Number of statements in base component member

CMPNT BASE LL

Component base level number

Component Delta Information

CMPNT DELTA MBR NAME

Component delta member name

CMPNT LAST LL DATE

Component delta date YYYY/MM/DD

CMPNT LAST LL TIME

Component delta time HH:MM:SS:TT

CMPNT LAST LL # STMTS

Number of statements in delta component member

CMPNT LAST LL # INSERT

Number of inserts in last level

CMPNT LAST LL # DELETE

Number of deletes in last level

CMPNT DELTA FORMAT

Delta format (F/R)

CMPNT LIST MONITOR

Component list built by monitor flag (M or blank)

CMPNT LIST COPIED

Component list copied or restored flag (C or blank)

CMPNT LIST LOC

Component list base stored in the delta library flag (D or blank)

Last Element Move Information

MOVE DATE

Element move date YYYY/MM/DD

MOVE TIME

Element move time HH:MM:SS:TT

MOVE USRID

User ID or job name associated with the move

Element Processor Information

PROC FLG

Processor flag (0/1/2/3/4)

PROC FAILED

Last processor failed flag (F or blank)

LAST PROC DATE

Last processor date YYYY/MM/DD

LAST PROC TIME

Last processor time HH:MM:SS:TT

GENERATE DATE

Last generate date YYYY/MM/DD

GENERATE TIME

Last generate time HH:MM:SS:TT

GENERATE USRID

User ID associated with last generate action

GENERATE CCID

CCID associated with last generate action

GENERATE COMMENT

Comment associated with last generate action

FAILED LIT

FAILED' - if last processor failed execution

PROC NAME

Name of the last processor executed

PROC RC

Processor return code

Last Element Retrieve Information

RETR DATE

Last retrieve date YYYY/MM/DD

RETR TIME

Last retrieve time HH:MM:SS:TT

RETR USRID

User ID associated with last retrieve action

RETR CCID

CCID associated with last retrieve action

RETR COMMENT

Comment associated with last retrieve action

Package Last Executed Against the Element Source Information

PKG ID (S)

Package ID executed against source

PKG EXEC DATE (S)

Date package executed against source YYYY/MM/DD

PKG EXEC TIME (S)

Time package executed against source HH:MM:SS:TT

Package Last Executed Against the Element Outputs Information

BACKED OUT (O)

The element outputs were created using a package and the outputs are currently backed out (Y or blank)

PKG ID (O)

Package ID executed against outputs

PKG EXEC DATE (O)

Date package executed against outputs YYYY/MM/DD

PKG EXEC TIME (O)

Time package executed against outputs HH:MM:SS:TT

Package/Element Locking Information

PKG ID

Element locked in package ID

LOCKED DATE

Date element locked YYYY/MM/DD

LOCKED TIME

Time element locked HH:MM:SS:TT

Last From CA Endevor SCM Location Information

SITE ID (F)

Site ID

ENV NAME (F)

Environment name

SYS NAME (F)

System name

SBS NAME (F)

Subsystem name

ELM (F)

Element name

TYPE NAME (F)

Type name

STG # (F)

Stage number (1/2)

ACT THAT UPDT (F)

Action that updated the from information

VV # (F)

Version number (00-99)

LL # (F)

Level number (001-999)

Retrieve to/Add/Updt from Information

ADD/UPDT DSN/PATH (F)

Add/Update from data set or USS path name

ADD/UPDT MBR/FILE (F)

ADD/Update from member or HFS file name

RETR DSN/PATH (T)

Retrieve-to data set or USS path name

RETR MBR/FILE (T)

Retrieve-to member or HFS file name

User Data Information

USER DATA

Element user data

Element Source Information

NOSOURCE

Y or N - Whether the element is a sourceless element. If Y, then the element is sourceless, meaning its element source and delta files are not present at this location.

BASE VV

Base version number

CMPNT BASE VV

Component base version number

Alter Action Metadata

ALT FIRST DATE

The date that the first Alter action was performed against this copy of the element. Values are formatted as yyyy:mm:dd or are blank.

ALT FIRST TIME

The time that the first Alter action was performed against this copy of the element. Values are formatted as hh:mm:ss:tt or are blank.

ALT LAST DATE

The date that the last Alter action was performed against this copy of the element. Values are formatted as yyyy:mm:dd or are blank.

ALT LAST TIME

The time that the last Alter action was performed against this copy of the element. Values are formatted as hh:mm:ss:tt or are blank. The value is blank if the Alter action was never run against this instance of the element. If the first date and time matches the last date and time, then the Alter action was only run once.

ALT LAST USERID

The user ID associated with the last Alter action performed against this copy of the element. Values are eight characters or are blank.

ALT CUMULATIVE CNT

The total number of times that Master Control File metadata fields were updated by *all* the Alter actions that were performed against this copy of the element. Values are five numerals or are blank.

ALT CCG

Whether the last Alter action updated the Generate CCID for this element. Possible values are Y or N. The value is blank if the Alter action was never run against this instance of the element.

ALT CCL

Whether the last Alter action updated the Last Action CCID for this element. Possible values are Y or N. The value is blank if the Alter action was never run against this instance of the element.

ALT CCR

Whether the last Alter action updated the Retrieve CCID for this element. Possible values are Y or N. The value is blank if the Alter action was never run against this instance of the element.

ALT DES

Whether the last Alter action updated the Description for this element. Possible values are Y or N. The value is blank if the Alter action was never run against this instance of the element.

ALT PRG

Whether the last Alter action updated the Processor Group for this element. Possible values are Y or N. The value is blank if the Alter action was never run against this instance of the element.

ALT SGN

Whether the last Alter action updated the Signout UserID for this element. Possible values are Y or N. The value is blank if the Alter action was never run against this instance of the element.

ALT USER DATA

Whether the last Alter action updated the User Data for this element. Possible values are Y or N. The value is blank if the Alter action was never run against this instance of the element.

Extracted Change Level Summary Data

This data is returned only when the option Element Change Level Summary or Component Change Level Summary is specified on the Data clause.

Element Level Information

RCD TYPE

S - The record type indicating this is a Change Level Summary record

DATA TYPE

E or C - The data type indicating this record contains either (E)lement or (C)omponent Change Level Summary data

SITE ID

Site ID

ENV NAME

Environment name

SYS NAME

System name

SBS NAME

Subsystem name

ELM NAME

Element name

TYPE

Type name

STG NAME

Stage name

STG ID

Stage ID

STG #

1 or 2 - Stage number

STG SEQ #

Relative Stage number

PROC GRP

Processor Group

UPDT DATE

Update Date

UPDT TIME

Update Time

SIGNOUT ID

Signout ID

ELM VV

Current Element Version

ELM LL

Current Element Level

CMPNT VV

Current Component Version

CMPNT LL

Current Component Level

RCD CNT

Number of Change Level Summary Records for this Element

Change Level Information

RCD #

0001 through ALELS_RS_RECCNT - Relative summary record number for this instance of an Element

CHG VV

Change Level Version

CHG LL

Change Level

CHG USRID

The user ID associated with this Change Level

CHG DATE

The date associated with this Change Level (yyyymmdd)

CHG TIME

The time associated with this Change Level (hhmmsstt)

 # STMTS

The number of statements contained in the Element at this Change Level

 # INSERT

The number of inserts associated with this Change Level

 # DELETE

The number of deletes associated with this Change Level

SYNC

C, S or blank - Indicates whether or not (blank) this Change Level is a (C)onsolidation Level or a (S)ynchronization Level

CCID

The CCID associated with this Change Level

COMMENT

The comment associated with this Change Level

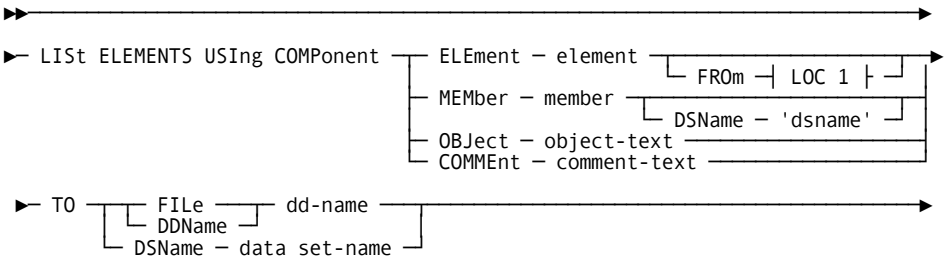
FULL ELM NAME

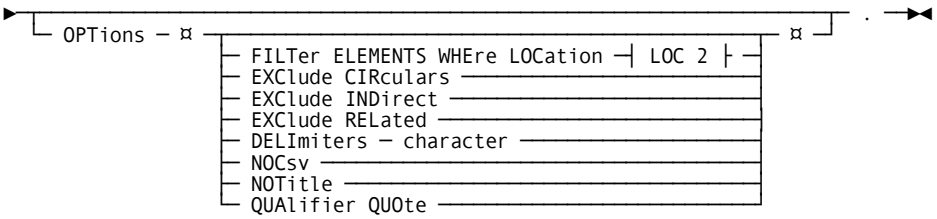
Element Long Name

The List Elements Using Component Function

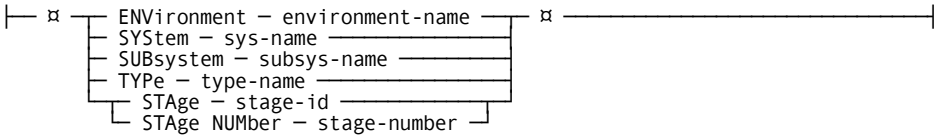
The list elements using component action lets you list the elements that use the component you specify.

This action has the following format:





Expansion of LOC 1 and LOC 2



Parameters

This action uses the common To clause and some of the common Option clause parameters. For more information, see Common To Clause and Common OPTIONS Keywords in the chapter "Using the Comma Separated Value (CSV) Utility" in the *Utilities Guide*. Other parameters used by this action include the following:

LISt ELEMENTS USIng COMPonent

Extracts a list of elements that use the specified component. You must limit the query to a component that is identified by one of the options. You can only code one of the following options:

ELEment element

Specifies a component that matches the specified 1- to 10-character element name. Name masking is allowed.

FROm *LOC 1*

(Optional) Specifies where the element is located.

MEMber *member*

Specifies a component that matches the specified 1- to 10-character member name. Name masking is allowed.

DSName '*dsname*'

(Optional) Specifies the 1- to 44-character data set name for the specified member. Enclose the data set name in single quotes. Name masking is allowed.

Object *object-text*

Specifies a component that matches the related 1- to 70-character object text string. Name masking is not allowed.

Comment *comment-text*

Specifies a component that matches the related 1- to 70-character comment text string. Name masking is not allowed.

OPTions

(Optional) Specifies various options. You can specify one or more of the options; the options are not mutually exclusive.

FILTer ELEMENTS WHEre LOCation *LOC 2*

Filters the ACM data to return only those elements found at the specified location.

EXClude CIRculars

Filters the ACM data to exclude elements that have a circular relationship to the object of your search.

EXClude INDirect

Filters the ACM data to exclude elements that are indirectly related to the object of your search.

EXClude RELated

Filters the ACM data to exclude elements that are related components.

LOC 1 and LOC 2

The location fields further qualify the element query object. One or more of the location parameters must be specified. You can specify the location by the environment, system, subsystem, or type. You can also specify stage ID or stage number, but not both.

ENVironment *environment-name*

Specifies the one- to eight-character name of the environment in which you want to perform your query.

SYStem *system-name*

Specifies the one- to eight-character name of the system in which you want to perform your query.

SUBsystem *subsystem-name*

Specifies the one- to eight-character name of the subsystem in which you want to perform your query.

TYPe *type-name*

Specifies the type of the elements for which you are searching. The type name can be one to eight characters in length.

STAge *stage-id*

Specifies the stage ID for the stage in which you want to perform your query.

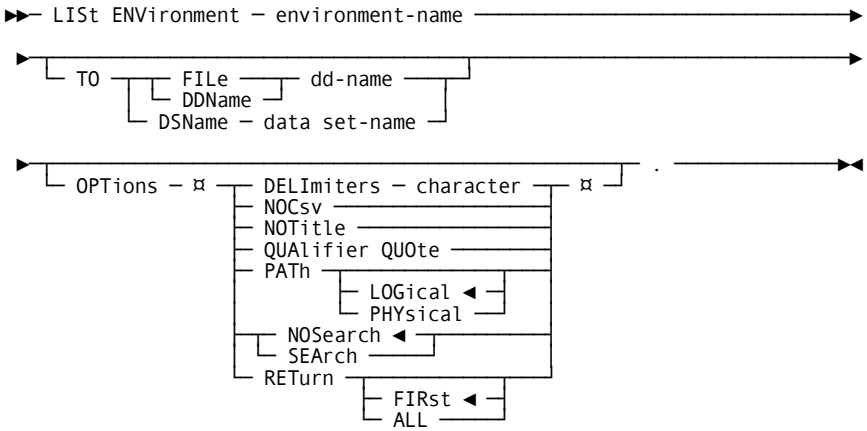
STAge NUMber *stage-number*

Specifies the number of the stage in which you want to perform your query.

The List Environment Function

The list environment CSV function allows you to extract environment information from the MCF that satisfies the criteria you specify.

The syntax of the list environment CSV function is shown next:



Wildcard and/or placeholder characters are permitted in the environment name field. On a non-explicit environment name, all environments that match its masking criteria are returned in the order in which they appear in the C1DEFLTS table.

Default mapping options on non-explicit environment name requests are: PATH PHYSICAL, SEARCH and RETURN ALL. The PATH LOGICAL, NOSEARCH and RETURN FIRST clauses are invalid on non-explicit environment name requests. Default mapping options on explicit environment name requests are: PATH LOGICAL, RETURN FIRST and NOSEARCH.

To get a complete list of defined environments, use the wildcard character for the environment name value. To get a mapped environment list, specify the starting environment name and use the SEARCH, PATH LOGICAL and RETURN ALL options. An explicit environment request with PATH PHYSICAL only returns that environment's information. The RETURN and SEARCH/NOSEARCH options have no effect.

List Environment Example

```
LIST ENV * TO FILE CSVFILE.
```

If 2 environments (ENVA and ENVB) are defined in the C1DEFLTS table, environment ENVA data will appear first, followed by ENVB.

Extracted Environment Data

The output file contains the following types of information:

SITE ID

Site ID

ENV NAME

Environment name

TITLE

Title

USR SEC TBL

User security name table

RESRC SEC TBL

Resource security name table

SMF SEC

SMF security recording (Y/N)

SMF ACT

SMF action recording (Y/N)

SMF ENV

SMF environment recording (Y/N)

DB BRIDGE AVAIL

DB Bridge available (Y/N)

DB BRIDGE ACTIVE

DB Bridge active (Y/N)

DB BRIDGE OPT1

DB Bridge option 1 (Y/N)

DB BRIDGE OPT2

DB Bridge option 2 (Y/N)

DB BRIDGE OPT3

DB Bridge option 3 (Y/N)

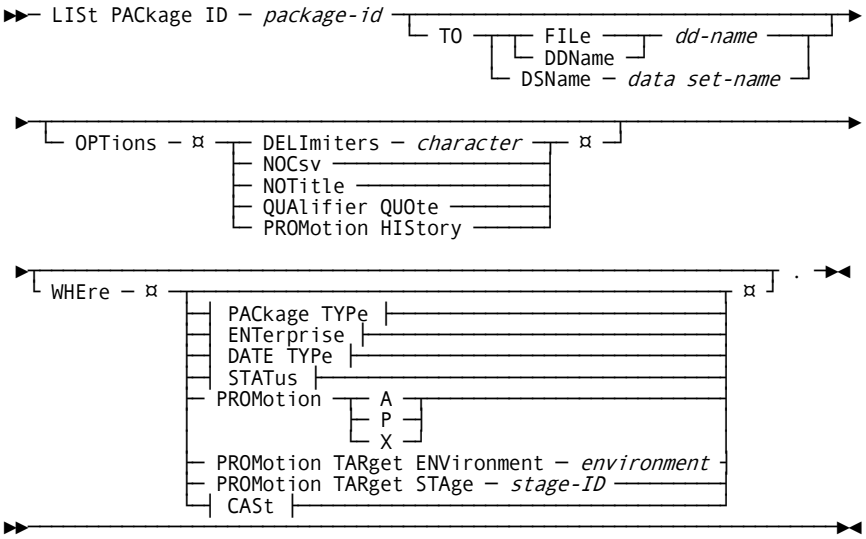
DB BRIDGE OPT4

DB Bridge option 4 (Y/N)

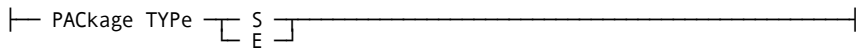
The List Package ID Function

The list package ID CSV function allows you to extract package information from the package file that satisfies the criteria you specify.

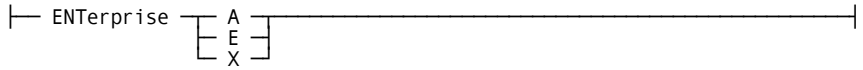
The syntax of the list package ID CSV function is shown next:



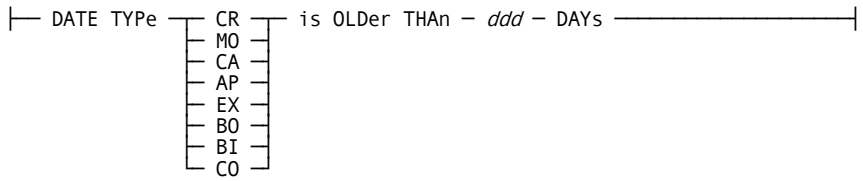
Expansion of PACKAGE TYPE



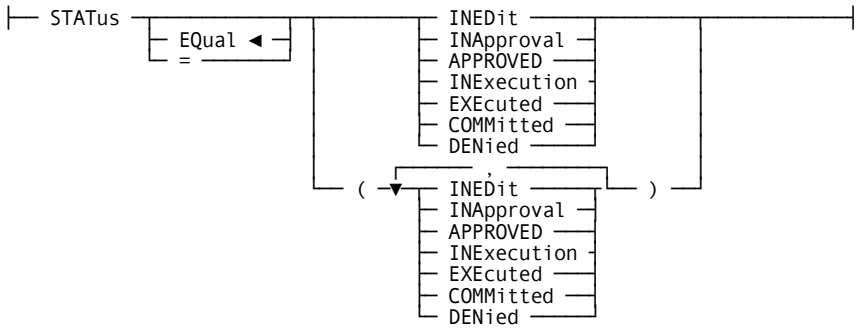
Expansion of ENTERPRISE



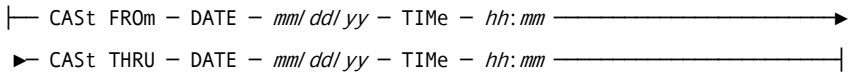
Expansion of DATE TYPE



Expansion of STATUS



Expansion of CAST



Name masking is permitted for the Package ID, Promotion Target Environment, and the Promotion Target Stage variables.

Parameters

The following parameters can be used with the list package ID function.

PROMotion

Specifies the promotion package type. Valid values are as follows:

A - Return promotion and non-promotion packages.

P - Return only promotion packages.

X - Exclude promotion packages; only return nonpromotion packages.

PROMotion TARget ENVironment *myenvir*

Specifies the promotion package target environment. This field only applies to promotion packages and can only be specified when the promotion package type is A or P. Name masking is supported. You can specify a target environment without specifying a target stage.

PROMotion TARget STAge *stage-ID*

Specifies the promotion package target stage ID. This field only applies to promotion packages and can only be specified when the promotion package type is A or P. Name masking is supported. You can specify a target stage without specifying a target environment.

Extracted Package Data

The output file contains the following types of information.

General Package Information

PKG ID

Package ID

SITE ID

Site ID

DESCRIPTION

Comment or description associated with this package

PKG TYPE

Package type. STANDARD or EMERGENCY

SHR OPT

Package shr option (Y/N)

BACKOUT FLG

Package backout enabled flag (Y/N)

WINDOW START DATE

Window start date DDMMMYY

WINDOW START TIME

Window start time HH:MM

WINDOW END DATE

Window end date DDMMMYY

WINDOW END TIME

Window end time HH:MM

STATUS

Package status. IN-EDIT, IN-APPROVAL, DENIED, APPROVED, IN-EXECUTION, EXECUTED, EXEC-FAILED or COMMITTED.

Package Create/Update Information

CREATE DATE

Date package was created DDMMMYY

CREATE TIME

Time package was created HH:MM

CREATE USRID

User ID associated with the create action

UPDT DATE

Date package header was last updated DDMMMYY

UPDT TIME

Time package header was last updated HH:MM

UPDT USRID

User ID associated with the last package header update

SCL DATE

Date package SCL was last modified DDMMMYY

SCL TIME

Time package SCL was last modified HH:MM

SCL USRID

User ID or jobname associated with the last SCL update

Package Correlation Information

CORR RCD EXIST

Correlation records exist flag (Y/N)

ENTERPRISE CORR

Enterprise correlation flag (Y/N) Indicates package is an enterprise package

 # CORR

Number of correlations associated with this package

Package Cast Information

CAST RCD EXIST

Cast records exist flag (Y/N)

CAST DATE

Date package was cast DDMMMYY

CAST TIME

Time package was cast HH:MM

CAST USRID

User ID associated with the cast action

Package Approval Information

APPR GRP EXISTS

Approver group records exist flag (Y/N)

APPR DATE

Date of final approval/denied DDMMMYY

APPR TIME

Time of final approval/denied HH:MM

Package Execution Information

EXEC BEGIN DATE

Package execution begin date DDMMMYY

EXEC BEGIN TIME

Package execution begin time HH:MM

EXEC END DATE

Package execution end date DDMMMYY

EXEC END TIME

Package execution end time HH:MM

EXEC USRID

User ID associated with the execution action

EXEC RC

Execution return code. Field remains blank until the package is executed.

ABNORMAL TERM

Execution of package abnormally terminated flag (Y/N)

Package Backout/Backin Information

BACKOUT N/A

Backout not in effect flag (Y/N)

BACKOUT RCD EXIST

Backout records exist flag (Y/N)

PKG BACKED OUT

Package has been backed out flag (Y/N), or element actions have been backed out (E).

PKG BACKOUT/BACKIN BEGUN

Package backout/backin has begun flag (Y/N)

PKG BACKOUT STATUS

Package backout status. Blank, BACKED-OUT, or ELMBACKOUT.

BACKOUT DATE

Date package was backed out DDMMMYY

BACKOUT TIME

Time package was backed out HH:MM

BACKOUT USRID

User ID associated with the backout action

BACKIN DATE

Date package was backed in DDMMMYY

BACKIN TIME

Time package was backed in HH:MM

BACKIN USRID

User ID associated with the backin action

Package Commit Information

COMMIT DATE

Date package was committed DDMMMYY

COMMIT TIME

Time package was committed HH:MM

COMMIT USRID

User ID associated with the commit action

Promotion Package Information

PROM PKG

Indicates if this is a promotion package. Possible values are Y, N, or blank.

PROM RCD EXIST

Indicates if promotion history records exist.

INT PKG ID

Internal package ID. The PKG ID field contains the external or original package ID. In the case of nonpromotion packages or the current version of a promotion package, these fields contain the same value. In the case of a historic version of a promotion package, this field contains the generated package name that is stored with the package.

PKG VERSION

Package version number. It is zero for nonpromotion packages. For promotion packages, it reflects the sequence of the package headers returned. The current version number is 1, the current-1 is 2, and so on. This is a calculated field; it is not stored on the package file.

PROM PREV PKGID

Previous promotion package ID. This field is populated in all promotion package headers, except the oldest (first) version.

PROM TGT ENV

Target location environment name associated with the move actions within a promotion package at the time of the last successful cast. This field is only populated for promotion packages.

PROM TGT STGID

Target location stage ID associated with the move actions within a promotion package at the time of the last successful cast.This field is only populated for promotion packages.

Miscellaneous Information

EXTERNAL SAF

External SAF flag (Y/N)

Note Information

NOTE

Notes, line 1

NOTE

Notes, line 2

NOTE

Notes, line 3

NOTE

Notes, line 4

NOTE

Notes, line 5

NOTE

Notes, line 6

NOTE

Notes, line 7

NOTE

Notes, line 8

ISO Dates and Times

UPDT DATE (ISO)

Date package header was last updated YYYY/MM/DD

UPDT TIME (ISO)

Time package header was last updated HH:MM:SS.TH

CREATE DATE (ISO)

Date package was created YYYY/MM/DD

CREATE TIME (ISO)

Time package was created HH:MM:SS.TH

SCL DATE (ISO)

Date package SCL was last modified YYYY/MM/DD

SCL TIME (ISO)

Time package SCL was last modified HH:MM:SS.TH

CAST DATE (ISO)

Date package was cast YYYY/MM/DD

CAST TIME (ISO)

Time Package was cast HH:MM:SS.TH

APPR DATE (ISO)

Date of final approval/denial YYYY/MM/DD

APPR TIME (ISO)

Time of final approval/denial HH:MM:SS.TH

WINDOW START DATE (ISO)

Execution window start date YYYY/MM/DD

WINDOW START TIME (ISO)

Execution window start time HH:MM:SS.TH

WINDOW END DATE (ISO)

Execution window end date YYYY/MM/DD

WINDOW END TIME (ISO)

Execution window end time HH:MM:SS.TH

EXEC BEGIN DATE (ISO)

Package execution begin date YYYY/MM/DD

EXEC BEGIN TIME (ISO)

Package execution begin time HH:MM:SS.TH

BACKOUT DATE (ISO)

Date package was backed out YYYY/MM/DD

BACKOUT TIME (ISO)

Time package was backed out HH:MM:SS.TH

BACKIN DATE (ISO)

Date package was backed in YYYY/MM/DD

BACKIN TIME (ISO)

Time package was backed in HH:MM:SS.TH

COMMIT DATE (ISO)

Date package was committed YYYY/MM/DD

COMMIT TIME (ISO)

Time package was committed HH:MM:SS.TH

EXEC END DATE (ISO)

Package execution end date YYYY/MM/DD

EXEC END TIME (ISO)

Package execution end time HH:MM:SS.TH

CA-7 SCHD DATE (ISO)

CA-7 schedule date YYYY/MM/DD

CA-7 SCHD TIME (ISO)

CA-7 schedule time HH:MM:SS.TH

User Data Information

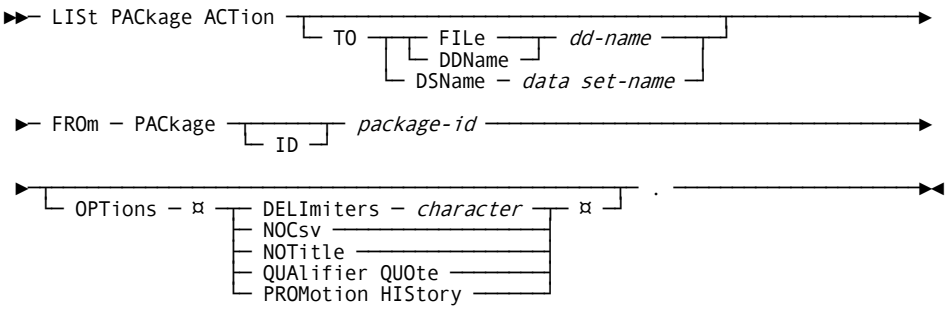
CVAL

Component validation flag: Y, N, or W (warn).

The List Package Action Summary Function

The list package action summary CSV function allows you to extract actions associated with a package from the package file that satisfies the criteria you specify.

The syntax of the list package action summary CSV function is shown next:



Name masking is permitted in the package-id field.

Extracted Package Action Summary Data

The output file contains the following types of information:

PKG ID

Package ID

SCL STMT#

SCL statement number

SITE ID

Site ID

ELM ACT

Element action (ADD, GENERATE, etc.)

ENDEVOR RC

Highest return code encountered

PROC RC

Highest processor return code encountered

BEGIN EXEC DATE

Package execution begin date YYYY/MM/DD

BEGIN EXEC TIME

Package execution begin time HH:MM

END EXEC DATE

Package execution end date YYYY/MM/DD

END EXEC TIME

Package execution end time HH:MM

CCID

CCID associated with this action

COMMENT

Comment associated with this action

UPDT DATE

Last physical record update date YYYY/MM/DD

UPDT TIME

Last physical record update time HH:MM

UPDT USRID

User ID or jobname associated with the last physical record update

APPR REQ (S)

Approval required flag (Y/blank)

VALIDATION REQ (S)

Validation required flag (Y/blank)

LOC (S)

Source location (C-CA Endevor SCM, F-File or DD name, D-Data set, A-CA Endevor SCM Archive, P-HFS Path)

SITE ID (S)

Site ID

ENV NAME (S)

Environment name

SYS NAME (S)

System name

SBS NAME (S)

Subsystem name

TYPE NAME (S)

Type name

STG # (S)

Stage number (1/2)

STG NAME (S)

Stage name

STG ID (S)

Stage ID

VVLL (S)

Version level (VVLL)

DELTA DATE

Last delta level date YYYY/MM/DD

DELTA TIME (S)

Last delta level time HH:MM

GENERATE DATE (S)

Last generate date YYYY/MM/DD

GENERATE TIME (S)

Last generate time HH:MM

PROC DATE (S)

Last processor date YYYY/MM/DD

PROC TIME (S)

Last processor time HH:MM

PREV PKG ID (S)

Previous package ID associated with source

APPR REQ (T)

Approval required flag (Y/blank)

VALIDATION REQ (T)

Validation required flag (Y/blank)

LOC (T)

Target location (C-CA Endevor SCM, F-File or DD name, D-Data set, A-CA Endevor SCM Archive, P-HFS Path)

SITE ID (T)

Site ID

ENV NAME (T)

Environment name

SYS NAME (T)

System name

SBS NAME (T)

Subsystem name

TYPE NAME (T)

Type name

STG # (T)

Stage number (1/2)

STG NAME (T)

Stage name

STG ID (T)

Stage ID

VVLL (T)

Version level (VVLL)

DELTA DATE

Last delta level date YYYY/MM/DD

DELTA TIME (T)

Last delta level time HH:MM

GENERATE DATE (T)

Last generate date YYYY/MM/DD

GENERATE TIME (T)

Last generate time HH:MM

PROC DATE (T)

Last processor date YYYY/MM/DD

PROC TIME (T)

Last processor time HH:MM

PREV PKG ID (T)

Previous package ID associated with target

ELM (S)

Source element name

DSN/PATH (S)

Source data set or USS path name

MBR/FILE (S)

Source member or HFS file name

ELM (T)

Target element name

DSN/PATH (T)

Target data set or USS path name

MBR/FILE (T)

Target member or HFS file name

INT PKG ID

Internal package ID. The PKG ID field contains the external or original package ID. In the case of nonpromotion packages or the current version of a promotion package, these fields contain the same value. In the case of a historic version of a promotion package, this field contains the generated package name that is stored with the package.

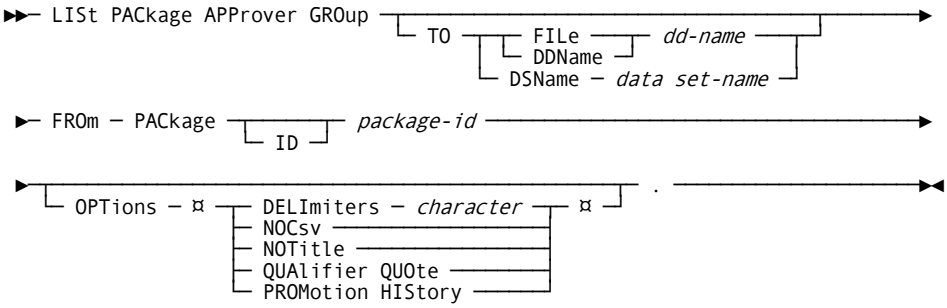
PKG VERSION

Package version number. It is zero for nonpromotion packages. For promotion packages, it reflects the sequence of the package headers returned. The current version number is 1, the current-1 is 2, and so on. This is a calculated field; it is not stored on the package file.

The List Package Approver Group Function

The list package approver group CSV function allows you to extract package approver group information from the package file that satisfies the criteria you specify.

The syntax of the list package approver group CSV function is shown next:



Name masking is permitted in the package-id field.

Extracted Package Approver Data

The output file contains the following types of information:

PKG ID

Package ID

APPR GRP NAME

Approver group name

ENV NAME

Approver group Environment

SITE ID

Site ID

OVERALL APPR STATUS

Overall approval status (blank, A or D)

APPR GRP TYPE

Approval group type (S-standard E-external)

QUORUM CNT

Quorum count. Minimum number of approvers that must approve this package.

SEQ#

One record is written for each approver in a group. This field contains an approver sequence number (1 of 16, 2 of 16, etc.).

 # APPR

Number of approvers in this group

UPDT DATE

Update date YYYY/MM/DD

UPDT TIME

Update time HH:MM

UPDT USRID

Update user ID

INT PKG ID

Internal package ID. The PKG ID field contains the external or original package ID. In the case of nonpromotion packages or the current version of a promotion package, these fields contain the same value. In the case of a historic version of a promotion package, this field contains the generated package name that is stored with the package.

PKG VERSION

Package version number. It is zero for nonpromotion packages. For promotion packages, it reflects the sequence of the package headers returned. The current version number is 1, the current-1 is 2, and so on. This is a calculated field; it is not stored on the package file.

Approver Information

USRID OF APPR

User ID of approver

APPR DATE

Approve/denial date YYYY/MM/DD

APPR TIME

Approve/denial time HH:MM

APPR STATUS

Approval status. Blank, APPROVED, DENIED.

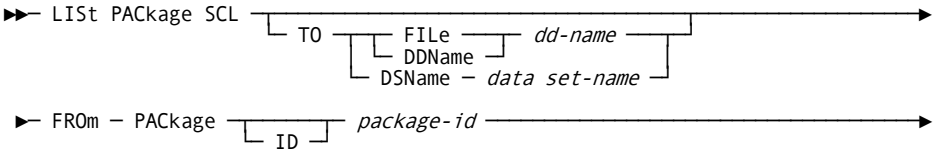
APPR REQ LIT

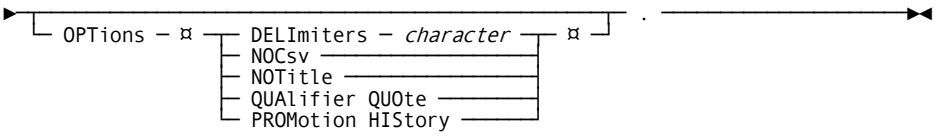
Approval required literal. Blank or REQUIRED. One output record is written for each approver associated with this group. All the data is duplicated in each record, except that the approver sequence number and approver information only appear in the record that is written for that particular approver.

The List Package SCL Function

The list package SCL CSV function allows you to extract package SCL statements associated from the package file that satisfies the criteria you specify.

The syntax of the list package SCL CSV function is shown next:





Name masking is permitted in the *package-id* field.

Extracted Package SCL Data

The output file contains the following types of information:

PKG ID

Package ID

SITE ID

Site ID

UPDT DATE

Date package SCL record was last modified YYYY/MM/DD

UPDT TIME

Time package SCL record was last modified HH:MM

UPDT USRID

User ID or jobname associated with the last SCL record update

 # STMTS

Number of SCL statements in this record. A maximum of 10 statements can exist in one record.

INT PKG ID

Internal package ID. The PKG ID field contains the external or original package ID. In the case of non-promotion packages or the current version of a promotion package, these fields contain the same value. In the case of an historical version of a promotion package, this field contains the generated package name that is stored with the package.

PKG VERSION

Package version number. It is zero for non-promotion packages. For promotion packages, it reflects the sequence of the package headers returned. The current version number is 1, the current-1 is 2, and so on. This is a calculated field; it is not stored on the package file.

SCL Information

SCL

SCL statement (1)

SCL

SCL statement (2)

SCL

SCL statement (3)

SCL

SCL statement (4)

SCL

SCL statement (5)

SCL

SCL statement (6)

SCL

SCL statement (7)

SCL

SCL statement (8)

SCL

SCL statement (9)

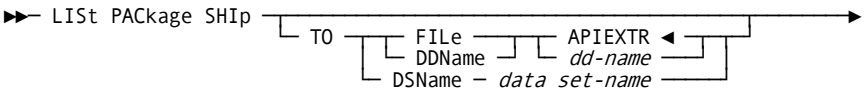
SCL

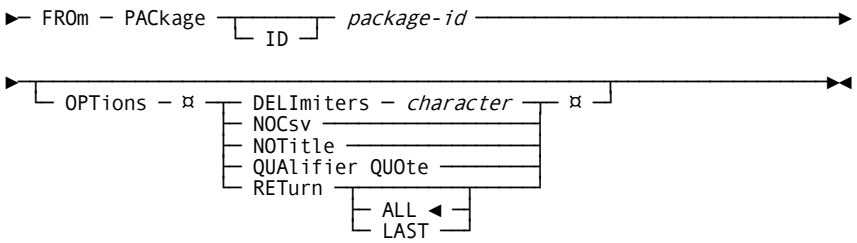
SCL statement (10) If any of the text found in the SCL statements contains the qualifier character (single quote or double quote), that character is converted to the qualifier character not in use. For example, if the qualifier character in use is a double quote and text in a CA Endevor SCM stored SCL statement is enclosed in double quotes, the CSV utility will enclose the text in single quotes instead of double quotes.

The List Package Ship Function

The list package ship SCL CSV function lets you to extract package ship data for specified packages.

The syntax of the list package ship SCL CSV function is shown next:





The entire TO clause is optional and defaults to DDName APIEXTR.

The FROm clause must be specified for the package ID of the package for which you want to obtain Package Shipment data. This field can be wildcarded.

The RETurn sub-option to the OPTions clause specifies how much data for each package is to be returned and defaults to ALL. The ALL option returns all package ship information associated with the specified package ID. The LASt option returns only the information associated with the last shipment of the specified package ID.

Note: Each shipment can contain up to ten destinations, resulting in up to ten output records for each shipment. This is why the SELECTED COUNT can be greater than the RETURNED COUNT in the API Execution Report. Multiple output records from the same shipment can be identified by the same PKG ID, SUBMIT DATE, and SUBMIT TIME fields.

Extracted List Package Ship Data

The output file contains the following types of information:

PKG ID

Package ID associated with this Package Shipment information

SUBMIT DATE

Shipment Create Date (ddMMMyy where MMM is the first 3 letters of the month)

SUBMIT TIME

Shipment Create Time (hh:mm)

SUBMIT usrid

The user ID or job name that created this Package Shipment request

UPDT DATE

The date this Shipment record was last updated (ddMMMyy)

UPDT TIME

The time this Shipment record was last updated (hh:mm)

RCD UPDT CNT

The number of times this record was updated

Pkg Ship Type

P or B - Whether the executed package (P) or the backed-out package (B) was shipped

CMPL Files

Y or N - Whether complementary files were shipped

CMPL FILES OVRD

O or blank - Whether complementary files were overridden

TRANS CODE

Transmission Method Code:

L

Local

X

XCOM

S

BDT1

B

BDT2

F

NVFT

M

CONN

Dest id

The Destination ID of the package

Host Stg RC

The return code of the host staging job

Host Trans RC

The return code of the host transmission job

Rmt Move RC

The return code of the remote move job

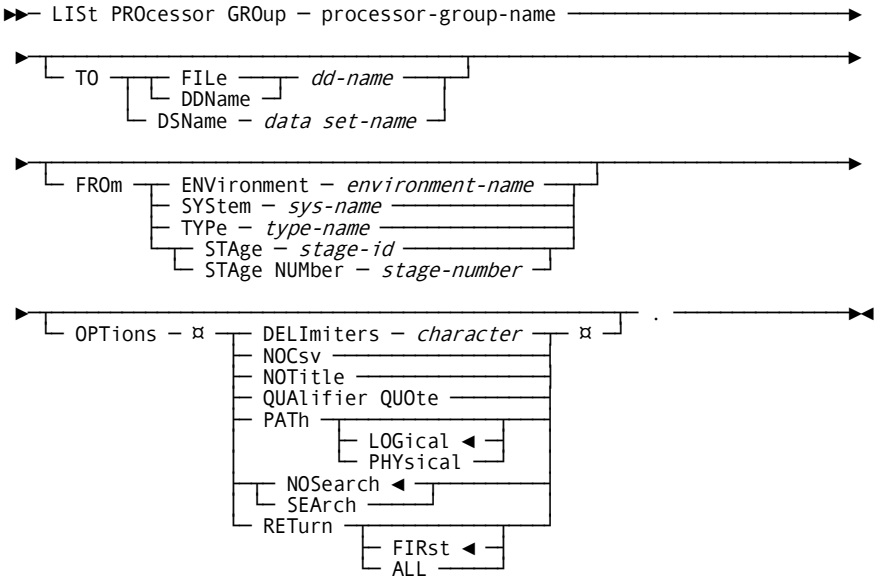
Rmt Job Name

The remote job name

The List Processor Group Function

The list processor group CSV function allows you to extract processor group information from the MCF that satisfies the criteria you specify.

The syntax of the list processor group CSV function is shown next:



Mapping is valid only when the environment name is explicitly specified. For explicit environment specification requests, processor group data is returned in map order. On non-explicit environments, processor group data is extracted based on environment and stage locations. Mapping between environment and stage locations is not supported on non-explicit environment requests.

The FROM clause is optional. All or part of the keyword specification on this clause can be omitted. On an omitted keyword, a wildcard is assumed. Also, when the environment name is wild, the STAge NUMber keyword cannot be used. You can either omit the stage keyword or specify it through the STAge stage-id keyword.

The PATH, SEARCH, NOSEARCH and RETURN clauses can be omitted. Default mapping options on non-explicit environment name requests are: PATH PHYSICAL, NOSEARCH and RETURN ALL. The PATH LOGICAL, SEARCH and RETURN FIRST clauses are invalid on non-explicit environment name requests. Default mapping options on explicit environment name requests are: PATH LOGICAL, RETURN FIRST and NOSEARCH. Explicit environment name requests for a single stage environment with no types defined to stage 1 require option NOSEARCH.

List Processor Group Examples

For the following two examples, two environments (ENVA and ENVB) are defined in the C1DEFLTS table where SYS1/SBS1/TYP1/TYP2 are defined to both environment. ENVA and ENVB are mapped together. Processor Group record GRP1 exists under SYS1/TYP1 of ENVB/S2. Processor Group record GRP2 exists under SYS1/TYP2 of both ENVA/S2 and ENVB/S2.

```
 LIST PRO GRP * TO FILE 'CSVFILE'
    FROM ENV'ENVA' SYS'*' TYP '*' STA'*'
    OPTIONS PATH LOGICAL SEARCH RETURN ALL.
```

Result for this example:

1. GRP1 from ENVB/S2/SYS1/TYP1
2. GRP2 from ENVA/S2/SYS1/TYP2
3. GRP2 from ENVB/S2/SYS1/TYP2

```
LIST PRO GRP * TO FILE 'CSVFILE'
   FROM ENV'*' SYS'*' TYP'*' STA'*'
-or-
LIST PRO GRP * TO FILE 'CSVFILE'.
```

Result for this example:

1. GRP2 from ENVA/S2/SYS1/TYP2
2. GRP1 from ENVB/S2/SYS1/TYP1
3. GRP2 from ENVB/S2/SYS1/TYP2

Extracted Processor Group Data

The output file contains the following types of information:

SITE ID

Site ID

ENV NAME

Environment name

SYS NAME

System name

TYPE NAME

Type name

STG NAME

Stage name

STG ID

Stage ID

STG #

Stage number (1/2)

PROC GRP NAME

Processor group name

PROC TYPE

Processor type (DEL/GEN/MOVE)

SYM OVRD #

Symbolic override number

UPDT DATE

Last physical record update date YYYY/MM/DD

UPDT TIME

Last physical record update time HH:MM:SS:TT

UPDT USRID

User ID or job name associated with last physical record update

DESCRIPTION

Processor group description

NEXT PROC GRP

Next processor group in map

PROC O/P TYPE

Processor output type

MOVE ACT PROC

Processor to use on MOVE action (G/M)

FG FLG

Foreground processing flag (Y/N)

PROC NAME

Processor name

TRNSFR ACT PROC

Processor to use on TRANSFER action (G/M)

Symbol Override Information

SYM OVRD LNG

Override symbol name length

SYM OVRD

Override symbol name value

SYM OVRD VALUE LNG

Override symbol value length

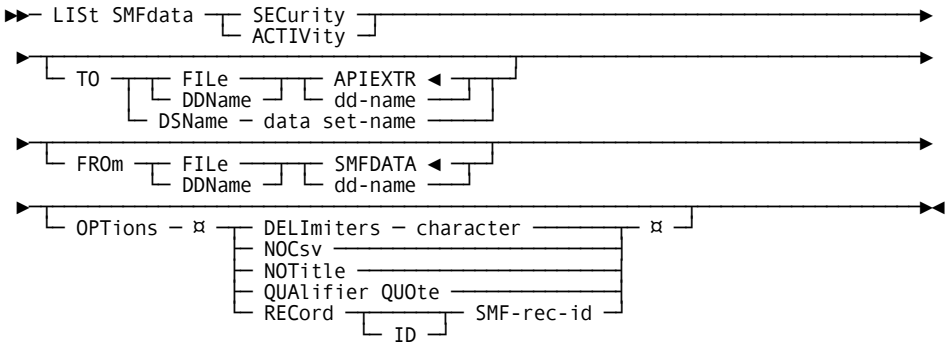
SYM OVRD VALUE

Override symbol value. One or more output records are written for each processor type (GEN, DEL or MOV). Also, one output record is written for each symbol found within a processor type. All the data except for the symbol override number (SYM OVRD #) and the symbol override information is duplicated in each subsequent record.

The List SMF Data Function

The list SMFdata CSV function lets you extract SMF record data corresponding to activity related to CA Endevor SCM actions or to security violations recorded by CA Endevor SCM.

The syntax of the list SMF data CSV function is shown next:



The SECurity and ACTIVity keywords are mutually exclusive.

The LIST SMFdata SECurity function provides information about CA Endevor SCM security violations.

The LIST SMFdata ACTIVity function provides information about activity performed against an element.

The entire FROm clause is optional and defaults to the ddname SMFDATA.

The TO clause is optional and defaults to ddname APIEXTR.

The OPTions clause with RECord ID allows you to provide the SMF record ID of the CA Endevor SCM SMF records for which you are requesting a list. The SMF record ID must be a number from 0-255. When not coded, it defaults to the SMF record ID (SMFREC#) specified in the defaults table C1DEFLTS.

Extracted SMF Activity Data

The output file contains the following types of information:

Header Information

RCD TYPE

A - SMF activity record.

SMF REC ID

SMF record ID.

ACT DATE

The date associated with this activity (yyyymmdd).

ACT TIME

The time associated with this activity (hhmmsstt).

CPU ID

The CPU ID associated with this activity.

ACT USRID

The user ID associated with this activity.

ACTION

The CA Endevor SCM action associated with this activity.

Environment Information

SITE

The Site ID associated with this activity.

ENV NAME

The source Environment name associated with this activity.

STG NAME

The source Stage name associated with this activity.

STG ID

The source Stage ID associated with this activity.

STG NUM

1 or 2 - The source Stage number associated with this activity.

SYS NAME

The source System name associated with this activity.

SBS NAME

The source Subsystem name associated with this activity.

TYPE NAME

The source Element Type associated with this activity.

ELM NAME

The source Element name associated with this activity.

PRGRP NAME

The processor group associated with this activity.

ELM VV

The source version of the Element associated with this activity.

ELM LL

The source level of the Element associated with this activity.

DATASET NAME

The dataset name associated with this activity.

MBR NAME

The dataset member name associated with this activity.

ENV NAME (T)

The target Environment name associated with this activity.

STG NAME (T)

The target Stage name associated with this activity.

STG ID (T)

The target Stage ID associated with this activity.

STG NUM (T)

1 or 2 - The target Stage number associated with this activity.

SYS NAME (T)

The target System name associated with this activity.

SBS NAME (T)

Th target Subsystem name associated with this activity.

TYPE NAME (T)

The target Element Type associated with this activity.

ELM NAME (T)

The target Element name associated with this activity.

ELM VV (T)

The target version of the Element associated with this activity.

ELM LL (T)

The target level of the Element associated with this activity.

Element Level Last Information

ELM LAST LL DATE

The Element Last Level Date at the time of this activity (yyyymmdd).

ELM LAST LL TIME

The Element Last Level Time at the time of this activity (hhmmssth).

ELM LAST LL USR

The user ID associated with the last level at the time of this activity.

ELM LAST LL ACT

The CA Endevor SCM Action associated with the last level at the time of this activity.

ELM LAST LL STMTS

The number of statements contained in the Element as of the last level at the time of this activity.

ELM LAST LL COMMENT

The Element Last Level Comment at the time of this activity.

Last Processor Information

LAST PROC NAME

The Processor name associated with the last processor execution at the time of this activity.

LAST PROC DATE

The date associated with the last processor execution at the time of this activity (yyyymmdd).

LAST PROC TIME

The time associated with the last processor execution at the time of this activity (hhmmssth).

LAST PROC USRID

The user ID associated with the last processor execution at the time of this activity.

LAST PROC RC

The return code associated with the last processor execution at the time of this activity.

LAST PROC NDVR RC

The CA Endevor SCM return code associated with the last processor execution at the time of this activity.

Request Parameter Information

CCID

The Change Control ID associated with this activity.

COMMENT

The Comment associated with this activity.

SIGNOUT OVRD

Y or N - Whether signout override was requested.

RETR COPY ONLY

Y or N - Whether the Retrieve action specified copy only (Retrieve action only).

EXPND INCL

Y or N - Whether the Retrieve action requested to expand includes (Retrieve action only).

REPLACE

Y or N - Whether the Retrieve action requested to replace an existing member (Retrieve action only).

ELM DEL

Y or N - Whether the element was deleted upon completion of the request.

IGN GEN FAILED

Y or N - Whether ignore generate failed was requested.

BYP GEN PROC

Y or N - Whether bypass generate processor was requested.

BYP DEL PROC

Y or N - Whether bypass delete processor was requested.

SYNC

Y or N - Whether the SYNC option was specified.

DEL ONLY CMPNT

Y or N - Whether to delete only components.

WITH HIST

Y or N - Whether this is a Move or Transfer With History.

ADD UPDT

Y or N - Whether this is an Add With Update.

PROC GRP OVRD

The name of the Processor Group when requested as an override.

SIGNOUT TO USR

The sign-out to user ID if specified.

Miscellaneous Additional Information

EXEC RC

The execution return code associated with this activity.

SRCHREPL FLAG

Y or N - Whether an update of either element source or component data by the Search and Replace utility was specfied.

AUTOGEN

Y or N - Whether Autogen was specified on the action. If AUTOGEN is Y, then AUTOGENX is N for that element.

AUTOGENX

Y or N - Whether Autogen was specified on the action for the where-used elements. If AUTOGENX is Y, then AUTOGEN is N for that element.

NOSOURCE

Y or N - Whether the NoSource option was specified on the Generate action.

AUTOGEN SPAN

N, A, S, or B – Whether SPAN NONE, SPAN ALL, SPAN SYSTEM, or SPAN SUBSYSTEM was specified for the AUTOGEN request.

FULL ELM NAME

The full element name associated with this activity.

Extracted SMF Security Data

The output file contains the following types of information:

RCD TYPE

S - SMF Security record.

SMF REC ID

The SMF record ID.

VIO DATE

The date of the security violation (yyyymmdd).

VIO TIME

The time of the security violation (hhmmsstt).

CPU ID

The CPU ID associated with the security violation.

VIO USRID

The user ID associated with the security violation.

ACTION

The CA Endevor SCM action associated with the security violation.

ERR CODE

The error code associated with the security violation.

ERR MESSAGE

The error message associated with the security violation.

SITE

The Site ID associated with the security violation.

ENV NAME

The Environment name associated with the security violation.

STG NAME

The Stage name associated with the security violation.

STG ID

The Stage ID associated with the security violation.

STG NUM

1 or 2 - The relative Stage number associated with the security violation.

SYS NAME

The System name associated with the security violation.

SBS NAME

The Subsystem name associated with the security violation.

TYPE NAME

The Element Type associated with the security violation.

ELM NAME

The Element name associated with the security violation.

PRGRP NAME

The processor group associated with the security violation.

DATASET NAME

The dataset name associated with the security violation.

MBR NAME

The dataset member name associated with the security violation.

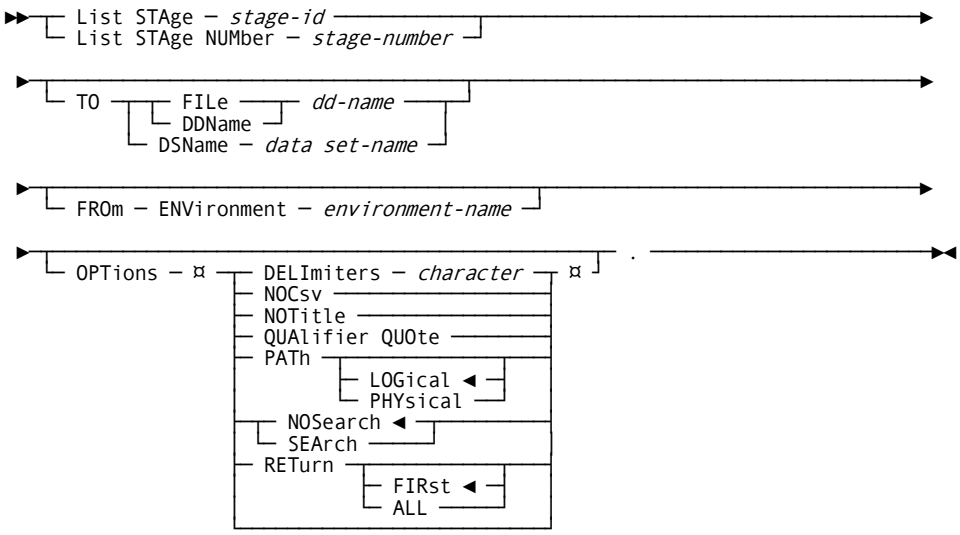
FULL ELM NAME

The full element name associated with the security violation.

The List Stage Function

The list stage CSV function allows you to extract stage information from the MCF that satisfies the criteria you specify.

The syntax of the list stage CSV function is shown next:



The wildcard character is permitted in the environment and stage number fields. On non-explicit environment name specifications, stage data information is extracted individually from each environment that meets its selection criteria. The FROM clause is optional. On an omitted environment name, a wildcard is assumed. Also, when the environment name is wild, the List STAge NUMber request cannot be used. You must use the LIST STAge request.

The PATH, SEARCH, NOSEARCH and RETURN clauses can be omitted. Default mapping options on non-explicit environment name requests are: PATH PHYSICAL, NOSEARCH and RETURN ALL. The PATH LOGICAL, SEARCH and RETURN FIRST clauses are invalid on non-explicit environment name requests. Default mapping options on explicit environment name requests are: PATH LOGICAL, RETURN FIRST and NOSEARCH.

List Stage Examples

For these examples, three environments (ENVA, ENVB and ENVC) are defined in the C1DEFLTS table. ENVA Stage 2 is mapped to ENVB at stage 2.

```
LIST STA * TO FILE CSVFILE
   FROM ENV '*'.
```

Result for this example:

1. Stage 1 data from ENVA
2. Stage 2 data from ENVA
3. Stage 1 data from ENVB
4. Stage 2 data from ENVB
5. Stage 1 data from ENVC
6. Stage 2 data from ENVC

```
LIST STA * TO FILE CSVFILE
   FROM ENV 'ENVA'
   OPTIONS PATH LOGICAL SEARCH RETURN ALL.
```

Result for this example:

1. Stage 1 data from ENVA
2. Stage 2 data from ENVA
3. Stage 2 data from ENVB

Extracted Stage Data

The output file contains the following types of information:

SITE ID

Site ID

ENV NAME

Environment name

STG NAME

Stage name

STG ID

Stage ID

STG #

Stage number (1/2)

TITLE

Title associated with stage

MCF DSN

Name of MCF file where stage is defined

ENTRY STG

Entry stage attribute (Y/N)

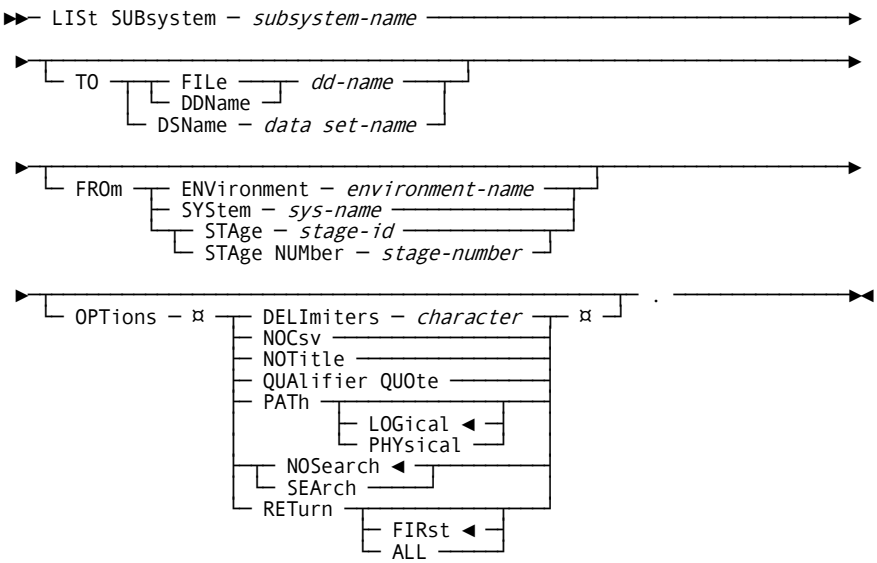
STOP@STG

Promotion package processing stop-at-stage (Y/N)

The List Subsystem Function

The list subsystem CSV function allows you to extract subsystem information from the MCF that satisfies the criteria you specify.

The syntax of the list subsystem CSV function is shown next:



A wildcard and/or placeholder character can be used in the environment, system, subsystem and stage fields. On non-explicit environment name specifications, subsystem data is extracted individually from each location (environment/stage) that meets its seleciton criteria.

The FROM ENVIRONMENT SYSTEM STAGE clause is optional. All or part of the keyword specification on this clause can be omitted. On an omitted keyword, a wildcard is assumed. Also, when the environment name is wild, the STAge NUMber keyword cannot be used. You can either omit the stage keyword or specify it through the STAge stage-id keyword.

The PATH, SEARCH, NOSEARCH and RETURN clauses can be omitted. Default mapping options on non-explicit environment name requests are: PATH PHYSICAL, NOSEARCH and RETURN ALL. The PATH LOGICAL, SEARCH and RETURN FIRST clauses are invalid on non-explicit environment name requests. Default mapping options on explicit environment name requests are: PATH LOGICAL, RETURN FIRST and NOSEARCH.

List Subsystem Examples

For the following two examples, two environments (ENVA and ENVB) are defined in the C1DEFLTS table where SYS1/SBS1/SBS2 are defined to both environment. ENVA Stage 2 is mapped to ENVB Stage 2.

```
LIST SUB * TO FILE 'CSVFILE'
   FROM ENV'*' SYS '*' STA'*'.
```

Result for this example:

1. SBS1 from ENVA Stage 1 under SYS1
2. SBS2 from ENVA Stage 1 under SYS1
3. SBS1 from ENVA Stage 2 under SYS1
4. SBS2 from ENVA Stage 2 under SYS1
5. SBS1 from ENVB Stage 1 under SYS1
6. SBS2 from ENVB Stage 1 under SYS1
7. SBS1 from ENVB Stage 2 under SYS1
8. SBS2 from ENVB Stage 2 under SYS1

```
LIST SUB * TO FILE 'CSVFILE'
    FROM ENV'ENVA' SYS '*' STA'*'
    OPTIONS RETURN ALL PATH LOGICAL SEARCH.
```

Result for this example:

1. SBS1 from ENVA Stage 1 under SYS1
2. SBS1 from ENVA Stage 2 under SYS1
3. SBS1 from ENVB Stage 2 under SYS1
4. SBS2 from ENVA Stage 1 under SYS1
5. SBS2 from ENVA Stage 2 under SYS1
6. SBS2 from ENVB Stage 2 under SYS1

Extracted Subsystem Data

The output file contains the following types of information:

SITE ID

Site ID

ENV NAME

Environment name

SYS NAME

System name

SBS NAME

Subsystem name

STG NAME

Stage name

STG ID

Stage ID

STG SEQ #

Relative stage sequence number

RCD UPDT CNT

Record update count

UPDT DATE

Update date YYYY/MM/DD

UPDT TIME

Update time HH:MM:SS:TH

UPDT USRID

Update user ID

TITLE

Subsystem Title

NEXT SBS

Next subsystem name in map

REL ID

Record created release ID

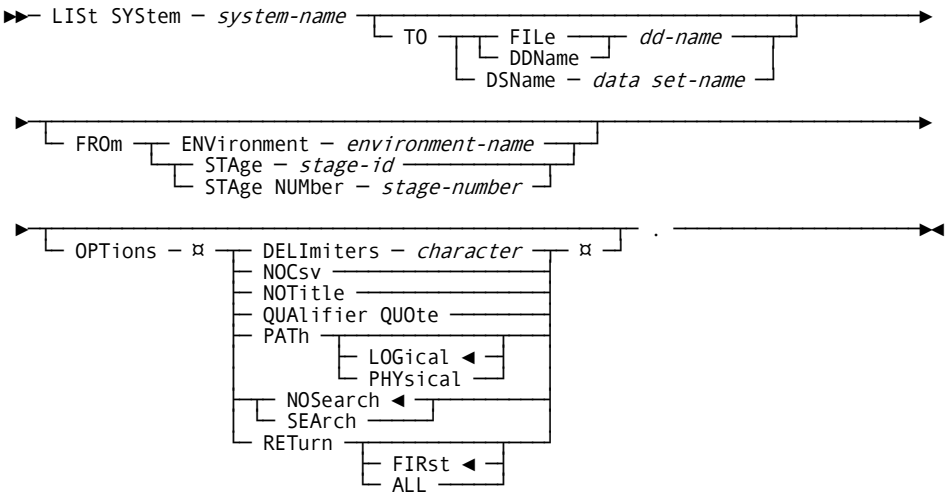
EXCLUDE FROM PROC O/P CK

Exclude Subsystem from the duplicate element processor output Type check (Y/N)

The List System Function

The list system CSV function allows you to extract system information from the MCF that satisfies the criteria you specify.

The syntax of the list system CSV function is shown next:



A wildcard and/or placeholder character can be used in the environment, system and stage fields. On non-explicit environment name specifications, system data will be extracted individually from each location (environment/stage) that meets its selection criteria.

The FROM ENVIRONMENT STAGE clause is optional. All or part of the keyword specification on this clause can be omitted. On an omitted keyword, a wildcard is assumed. Also, when the environment name is wild, the STAge NUMber keyword cannot be used. You can either omit the stage keyword or specify it through the STAge stage-id keyword.

The PATH, SEARCH, NOSEARCH and RETURN clauses can be omitted. Default mapping options on non-explicit environment name requests are: PATH PHYSICAL, NOSEARCH and RETURN ALL. The PATH LOGICAL, SEARCH and RETURN FIRST clauses are invalid on non-explicit environment name requests. Default mapping options on explicit environment name requests are: PATH LOGICAL, RETURN FIRST and NOSEARCH.

List System Examples

For the following two examples, two environments (ENVA and ENVB) are defined in the C1DEFLTS table where SYS1/SYS2 are defined to both environment. ENVA Stage 2 is mapped to ENVB Stage 2.

```
LIST SYS * TO FILE 'CSVFILE'
   FROM ENV'*' STA'*'.
```

Result for this example:

1. SYS1 from ENVA Stage 1
2. SYS2 from ENVA Stage 1
3. SYS1 from ENVA Stage 2
4. SYS2 from ENVA Stage 2
5. SYS1 from ENVB Stage 1
6. SYS2 from ENVB Stage 1
7. SYS1 from ENVB Stage 2
8. SYS2 from ENVB Stage 2

```
LIST SYS * TO FILE 'CSVFILE'
    FROM ENV 'ENVA' STA'*'
    OPTIONS RETURN ALL PATH LOGICAL SEARCH.
```

Result for this example:

1. SYS1 from ENVA Stage 1
2. SYS1 from ENVA Stage 2
3. SYS1 from ENVB Stage 2
4. SYS2 from ENVA Stage 1
5. SYS2 from ENVA Stage 2
6. SYS2 from ENVB Stage 2

Extracted System Data

The information contained in the output file is explained in the following list:

SITE ID

Site ID

ENV NAME

Environment name

SYS NAME

System name

STG NAME

Stage name

STG ID

Stage ID

STG SEQ #

Relative stage sequence number

RCD UPDT CNT

Record update count

UPDT DATE

Update date YYYY/MM/DD

UPDT TIME

Update time HH:MM:SS:TH

UPDT USRID

Update user ID

TITLE

Subsystem Title

NEXT SYS

Next system name in map

PROC LOAD LIB

Processor load library name

PROC LIST LIB

Processor listing library name

COMMENT REQ

Comment required flag (Y/N)

CCID REQ

CCID required flag (Y/N)

SIGNOUT REQ

Signout required flag (Y/N)

VALIDATE RETR DSN

Validate retrieve to data set name (Y/N)

JUMP OPT REQ

Jump location allowed flag (Y/N)

BACKUP DATE

Last backup date YYYY/MM/DD

BACKUP TIME

Last backup time HH:MM:SS:TT

REL ID

Record created release ID (nn)

DUP ELM REG

Duplicate element registration (Y/N)

DUP ELM REG SEV LL

Duplicate element registration message severity level (W,C,E,S).

This field will be blank if DUP ELM REG is set to a value of N.

PROC O/P REG

Processor output type registration (Y/N)

PROC O/P REG SEV LL

Processor output type registration message severity level (W,C,E,S).

This field will be blank if PROC O/P REG is set to a value of N.

PROC O/P REG ACROSS SBS

Duplicate element processor output Type check across Subsystems (Y/N)

ELM LL AGE RETENTION

Retain element levels by age (Y/N)

CMPNT LL AGE RETENTION

Retain Component levels by age (Y/N)

RETAIN ELM LL # MTH

Retain element levels for # of months

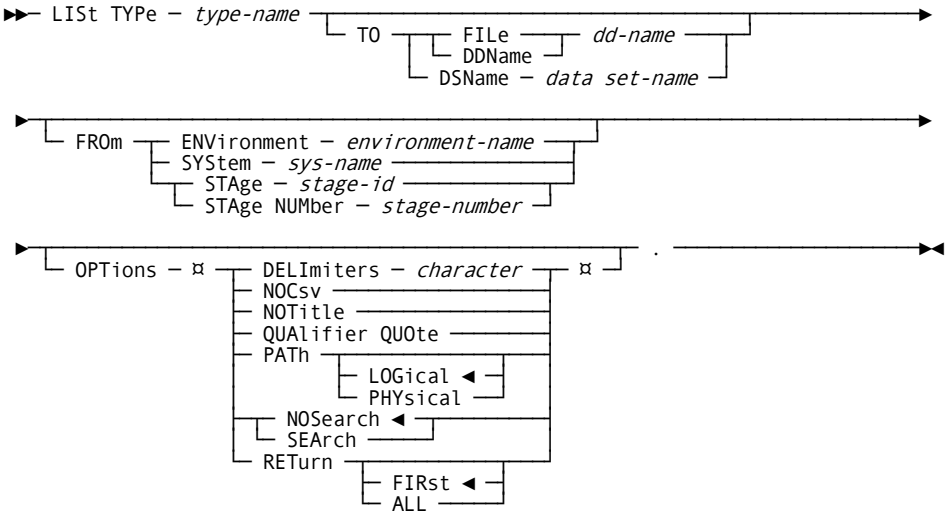
RETAIN CMPNT LL #MTH

Retain component levels for # of months

The List Type Function

The list type CSV function allows you to extract type information from the MCF that satisfies the criteria you specify.

The syntax of the list type CSV function is shown next:



Wildcard and/or placeholder characters are permitted in the environment, system, stage and type fields. On non-explicit environment name specifications, type data will be extracted individually from each location (environment/stage) that meets its selection criteria.

The FROM clause is optional. All or part of the keyword specification on this clause can be omitted. On an omitted keyword, a wildcard is assumed. Also, when the environment name is wild, the STAge NUMber keyword cannot be used. You can either omit the stage keyword or specify it through the STAge stage-id keyword.

The PATH, SEARCH, NOSEARCH and RETURN clauses can be omitted. Default mapping options on non-explicit environment name requests are: PATH PHYSICAL, NOSEARCH and RETURN ALL. The PATH LOGICAL, SEARCH and RETURN FIRST clauses are invalid on non-explicit environment name requests. Default mapping options on explicit environment name requests are: PATH LOGICAL, RETURN FIRST and NOSEARCH. Explicit environment name requests for a single stage environment with no types defined to stage 1 require option NOSEARCH.

List Type Examples

For the following two examples, two environments (ENVA and ENVB) are defined in the C1DEFLTS table where SYS1/TYP1/TYP2 are defined to both environments. ENVA Stage 2 is mapped to ENVB Stage 2.

```
LIST TYP * TO FILE 'CSVFILE'
   FROM ENV '*' SYS '*' STA '*'.
```

Result for this example:

1. TYP1 from ENVA Stage 1 under SYS1
2. TYP2 from ENVA Stage 1 under SYS1
3. TYP1 from ENVA Stage 2 under SYS1
4. TYP2 from ENVA Stage 2 under SYS1
5. TYP1 from ENVB Stage 1 under SYS1
6. TYP2 from ENVB Stage 1 under SYS1
7. TYP1 from ENVB Stage 2 under SYS1
8. TYP2 from ENVB Stage 2 under SYS1

```
FROM ENV' ENVA' SYS '*' STA '*'
      OPTIONS RETURN ALL PATH LOGICAL SEARCH.
```

Result for this example:

1. TYP1 from ENVA Stage 1 under SYS1
2. TYP1 from ENVA Stage 2 under SYS1
3. TYP1 from ENVB Stage 2 under SYS1
4. TYP2 from ENVA Stage 1 under SYS1
5. TYP2 from ENVA Stage 2 under SYS1
6. TYP2 from ENVB Stage 2 under SYS1

Extracted Type Data

The output file contains the following types of information:

SITE ID

Site ID

ENV NAME

Environment name

SYS NAME

System name

TYPE NAME

Type name

TYPE # ID

Type number ID

STG NAME

Stage name

STG ID

Stage ID

STG #

Stage number (1/2)

STG SEQ #

Relative stage sequence number

RCD UPDT CNT

Record update count

UPDT DATE

Update date YYYY/MM/DD

UPDT TIME

Update time HH:MM:SS:TH

UPDT USRID

Update user ID

REL ID

Record created release ID

NEXT TYPE

Next type name in map

DESCRIPTION

Type description

DFLT PROC GRP

Default processor group name

DATA FORMAT

Format of this type of data. B-Binary, T-TEXT, or blank-not specified.

FILE EXT

File extension associated with this type of data. Valid values for this 8-character field are: a-z, A-Z, 0-9, or blanks. Trailing blanks are allowed, but embedded blanks are not.

LANG

Language (COBOL, DATA, etc.)

PV/LB LANG

PV/LB language

REGR %

Regression percent

REGR SEV

Regression error severity level (W/C/E/S)

SRC LNG

Maximum length of source

COMPARE (F)

Start compare column

COMPARE (T)

End compare column

AUTO CONSOL

Auto consolidate flag (Y/N)

CONSOL LL

Consolidation level (000-999)

AUTO CONSOL LL

Auto consolidation level

CMPNT AUTO CONSOL

Component auto consolidate flag (Y/N)

CMPNT CONSOL LL

Component consolidation level (000-999)

CMPNT AUTO CONSOL LL

Component auto consolidation level

EXPAND INCL

Expand includes flag in source output library (Y/N)

FWD/REV/IMG/LOG ELM DELTA

Forward/Reverse/Image/Log delta for element (F/R/I/L)

FWD/REV CMPNT DELTA

Forward/Reverse delta for components (F/R)

COMPRESS BASE

Compress base flag (Y/N)

ELM NAME NOT ENCRYPTED

Not encrypted element name (Y/N)

SRC O/P DS TYPE

Source output data set type (PO/PV/LB/??)

SRC O/P DSN

Source output data set name

INCL DS TYPE

Include data set type (PO/PV/LB/??)

INCL DSN

Include data set name

BASE DS TYPE

Base data set type (PO/PV/LB/EL/VK/??)

BASE/IMAGE DSN

Base data set name

DELTA DS TYPE

Delta data set type (PO/PV/LB/EL/VK/??)

DELTA DSN

Delta data set name

HFS DELIMITER

HFS file delimiter (COMP/CR/CRLF/F/LF/NL/V) The source output, include, base, and delta data set names are displayed in the format in which they are defined. Therefore, symbolic values are not expanded. The data set type fields will contain a value of ?? (unknown) if the corresponding data set name contains symbolic values.

ELEMENT RECFM

Record length format (F/V/N)

CSV Utility JCL

Sample JCL to execute the CSV utility is located in *iprfx.iqual*.CSIQJCL, member name BC1JCSVU.

Alternatively, the CSV utility can be loaded from a non-authorized library, which may be preferred if you are executing this utility from within a processor. To call the CSV utility using a non-authorized program, specify the following:

EXEC PGM=BC1PCSV0

To use this EXEC statement, the release CONLIB must be in the STEPLIB concatenation. In addition, users executing the non-authorized program must have the requisite security access to the CA Endevor SCM data sets for the function requested. The caller's security profile must have sufficient CA Top Secret, CA ACF2, or RACF, access to the CA Endevor SCM control files (catalog, master, package, base, delta), because the CA Endevor SCM alternate ID facility is not available when running in an unauthorized mode.

The CSV utility shuts down the API server after the utility is finished. To disable shutdown of the API server, so that the next API call will not start the API server from the beginning, the parameter RETAINAPI must be passed to the JCL or REXX program. For example, to call the CSV utility with the parameter from a REXX program, use the following:

```
SELECT PGM(BC1PCSV0) PARM(RETAINAPI)
```

At the end of the calling program, the CSV utility must shut down the API, otherwise the program causes an SA03 abend. So the last call of BC1PCSV0 utility must be done without the RETAINAPI parameter.

Under some circumstances, CA Endevor SCM automatically ignores the RETAINAPI parameter to prevent the SA03 abend. This is done when BC1PCSV0 is attached by the program. For example, the utility is attached by a program when it is used directly in the JCL as PGM=BC1PCSV0,PARM=RETAINAPI or it is used in a REXX program running from IKJEFT01 in the JCL.

For more information about the API, see the *API Reference Guide*.

Chapter 14: Using the Age-Managed Delta Utility

This section contains the following topics:

Age-Managed Delta Utility

Setting the age retention limits in the system definitions, to indicate how long delta levels will be kept, turns on the Age Level Retention feature. 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.

Note: Delta removal will occur at the first modification of the elements or components after turning on the Age Level Retention feature. For more information, see Implement Age Level Retention in the *Administration Guide*.

Identify Old Delta Levels

You can identify aged element and component levels by running the BC1JRDLT utility. To identify old delta levels, edit JCL member BC1JRDLT in your installation's CSIQJCL data set to conform to your installation's standards, and then submit BC1JRDLT. The utility's output report will list all the elements and components that are older than the age limits set in the system definitions.

Chapter 15: Using the Unused Processor Symbolic Overrides Utility

This section contains the following topics:

Unused Processor Symbolic Overrides Utility

The BC1PSDEL utility lets you list and remove unused processor symbolic overrides. Symbolic overrides are identified as unused when the corresponding symbolic no longer exists in a processor.

If you updated a processor and removed some symbolics from its PROC statement, the corresponding symbolic overrides defined for that processor would remain in the Master Control File. These are considered to be unused symbolic overrides, because the corresponding symbolic is removed from the processor for which the override existed.

If you later modify the processor to include the same symbolic, then the override becomes active again without the user's awareness. This can cause unexpected behavior of the changed processor. Also, the API and the CSV utility report unused symbolic overrides, without identifying them as unused. To avoid these situations, use the BC1PSDEL utility to identify and remove any unused symbolic overrides. The utility cannot prevent symbolics from becoming unused; it can identify them for you and remove them.

The BC1PSDEL utility scans all Master Control Files searching for unused symbolic overrides. It does not directly remove them, but does produce output with "DELETE PROCESSOR SYMBOL" SCL statements, which can be reviewed and passed to the Batch Environment Administration facility (ENBE1000) to be processed.

The utility identifies and classifies unused processor symbolic overrides as follows:

- Regular unused symbolic overrides, which originate from a source change of the corresponding processor.
- Irregular unused symbolic overrides, which are identified if the corresponding processor load module or load library does not exist or cannot be accessed. In this case, the utility treats all respective symbolic overrides as unused, because it cannot be determined if the corresponding symbolics exist in the processor load module.

List and Remove Unused Processor Symbolic Overrides

Run the JCL in member BC1JSDEL found in your iprfx.iqual.CSIQJCL library to list and remove any unused processor symbolic overrides. The JCL will execute the BC1PSDEL utility to list the unused symbolic overrides, build SCL statements to remove them, and optionally execute the Batch Environment Administration facility (ENBE1000) to delete the unused overrides. The utility can be run on a scheduled basis or as necessary.

To list and remove unused processor symbolic overrides

1. Modify the BC1JSDEL JCL member as follows:
   a. Add a valid job card.
   b. Set the execution mode parameter. Choose one of the following:

      MODE =1

      - If the utility has been run before, this mode deletes previously created SCL data sets, before building new SCL data sets.
      - Executes the BC1PSDEL utility to build SCL data sets, but does **not** process them.

      MODE=2

      - If the utility has been run before, this mode deletes previously created SCL data sets, before building new SCL data sets.
      - Executes the BC1PSDEL utility to build SCL data sets.
      - Executes the Batch Environment Administration facility (ENBE1000) to process the SCL statements.

      MODE=3

      - Executes the Batch Environment Administration facility (ENBE1000) to process the SCL statements built in the previous execution of this job.

   c. Make sure the DD statements refer to the correct SCL1 and SCL2 data sets.

2. Run the BC1JSDEL JCL.

   The results depend on the mode you selected. Modes 1 and 2 identify the unused symbolic overrides and create SCL to delete them, but only Modes 2 and 3 delete the unused symbolic overrides.

   MODE 1 or MODE 2

   If the job runs in MODE 1 or 2, the BC1PSDEL utility scans all Master Control Files for unused processor symbolic overrides and generates the following output.

   Note: MODE 2 also executes the SCL statements, for details see section "MODE 2 or MODE 3".

   - REPORT - list of nonexistent/inaccessible processor load modules (if any) and some scanning statistics.
   - Lists unused symbolic overrides in the following data sets:

     SYMLIST1 - list of regular unused symbolic overrides

     SYMLIST2 - list of irregular unused symbolic overrides
   - Creates "DELETE PROCESSOR SYMBOL" SCL statements in the following data sets:

     SCL1 - for regular unused symbolic overrides

     SCL2 - for irregular unused symbolic overrides.
   - Return codes:

     RC=0 if no unused symbolic overrides are found.

     RC=4 if some regular, but no irregular unused symbolic overrides are found.

     RC=8 if some irregular unused symbolic overrides are found.

     Note: RC=8 indicates a symptom of another problem, which is that the corresponding processor load module or processor load library does not exist or is not accessible. In this case, the BC1PSDEL utility cannot determine if the symbolic overrides are valid, so it implicitly designates them as irregular unused symbolic overrides. We don't recommend deleting the irregular unused symbolic overrides, unless you know and understand why the corresponding load module(s) cannot be located or accessed. Therefore the SCL statements are split into two groups, so you can decide whether to delete the irregular unused symbolic overrides along with the regular ones. By default, SCL2 is not included in the Batch Environment Administration (ENBE1000) step. If you are sure you want to remove the irregular unused symbolic overrides, uncomment the SCL2 data set in the ENESCLIN DD statement.

MODE 2 or MODE 3

If the job runs in MODE 2 or 3, the Batch Environment Administration facility (ENBE1000) processes the "DELETE PROCESSOR SYMBOL" SCL statements generated by the BC1PSDEL utility and deletes the unused processor symbolic overrides from Master Control Files. By default, only SCL1 statements are executed. However, if you uncommented the SCL2 data set in the ENESCLIN DD statement, the SCL2 statements are executed also.

If any regular unused symbolic overrides are deleted, the Batch Environment Administration facility (ENBE1000) finishes with RC=4 and the action log contains ENBE144W message(s):

> ENBE144W Processor Symbol X has not been overridden or is not defined for Generate/Delete/Move processor Y.

This is correct because the Batch Environment Administration facility cannot see regular unused symbolic overrides; however, it really does remove them.

Chapter 16: Using the BSTXCOPY Utility

BSTXCOPY is a CA Endevor SCM utility that performs the same functions as the CA Endevor SCM processor utility BSTCOPY. However, you can use BSTXCOPY to perform these functions outside of a CA Endevor SCM processor. BSTCOPY can only be executed within a CA Endevor SCM processor.

BSTXCOPY can be used to copy between PDS load libraries and PDSE load libraries, from PDSE to PDSE, and from PDSE back to PDS. BSTXCOPY can copy aliases but the original loadlib member must be present in the From library.

The following JCL shows how to execute this utility. The first library in the STEPLIB contains the Endevor site-specific modules (C1DEFLTS, ENDICNFG, and so on):

```
//COPYEX EXEC PGM=BSTXCOPY
//STEPLIB DD DISP=SHR,DSN=YOUR.ENDEVOR.CSIQAUTU
//               DD DISP=SHR,DSN=YOUR.ENDEVOR.CSIQAUTH
//               DD DISP=SHR,DSN=YOUR.ENDEVOR.CSIQLOAD
//SYSPRINT DD SYSOUT=*
//SYSUT3   DD UNIT=VIO,SPACE=(CYL,(1,2))
//SYSUT4   DD UNIT=VIO,SPACE=(CYL,(1,2))
//IND1     DD DISP=(OLD,PASS),DSN=YOUR.PDS.LOADLIB   (PDS)
//IND2     DD DISP=(OLD,PASS),DSN=YOUR.PDS.LOADLIB  (PDS)
//OUTDD    DD DISP=(OLD,PASS),DSN=YOUR.NEW.PDSE.LOADLIB  (PDSE)
//SYSIN    DD *
   COPY INDD=((IND1,R)),OUTDD=OUTDD
   COPY INDD=((IND2,R)),OUTDD=OUTDD
```

For more information about the input syntax and restrictions, see The BSTCOPY Utility, in the chapter "Using Processor Utilities," in the *Extended Processor Guide*.

Index

F

G

I

J

L

M

N

P

R

S

U

V