CA Endevor® Software Change Manager

SCL Reference Guide

Version 17.0.00

Second Edition

CA Technologies Product References

This document references the following CA Technologies products:

CA Endevor® Software Change Manager (CA Endevor SCM)

CA Endevor® Software Change Manager Automated Configuration (CA Endevor Automated Configuration)

CA Change Manager Enterprise Workbench (CA CMEW)

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Documentation Changes

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

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

Version 17.0, Second Edition

- Updated the following topics to add the Alter action:
  - Set Options Syntax (see page 57)— Added the Alter action Search options: All and First Found
  - Actions and the Set Options Statement (see page 66)— Indicated which options are valid for the Alter action.
  - Element Action Statements (see page 41)
  - The Alter Statement (see page 95)
- Print Element Syntax (see page 154)— Updated to clarify that Print actions that use the Explode option are processed first before other actions are processed in Type sequence order.

Version 17.0

- Restyled the introductory information in the chapter "Managing Environments in Batch" into the following topics: How to Manage Environments in Batch (see page 242), Create SCL for Environment Objects (see page 244), Execute the Batch Environment Administration Facility (see page 245), and Batch Admin Edit Macros (see page 247).
- Updated various topics to indicate that you can name mask the From Environment *environment-name* value in certain Build SCL statements. The following topics were updated: Name-Masking (see page 23), Build Statements (see page 250), Build SCL for Approver Group Syntax (see page 250), Build SCL for Approver Relations Syntax (see page 252), Build SCL for Environment Syntax (see page 253), Build SCL for Processor Group Syntax (see page 255), Build SCL for Processor Symbol Syntax (see page 257), Build SCL for Subsystem Syntax (see page 259), Build SCL for System Syntax (see page 261), Build SCL for Type Syntax (see page 263), and Build SCL for Type Sequence Syntax (see page 264).

Version 16.0, Third Edition

For PTF RO67758, March 2014:

- List from Archive Data Set Syntax (see page 133)— Updated to describe how the Set From statement affects the from criteria.

Version 16.0, Second Edition

- Define System Syntax (see page 285)— Updated to correct the syntax diagram for the ACROSS SUBSYSTEMS option on the DUPLICATE PROCESSOR OUTPUT TYPE CHECK clause. The correct abbreviation for this option is ACROSS SUBS.

For PTF RO67758, March 2014:

- List from Archive Data Set Syntax (see page 133)— Updated to describe how the Set From statement affects the from criteria.

Version 16.0

- Set Options Syntax (see page 57), Print Element Syntax (see page 154)— Updated the NOCC option description.
- Set To Syntax (see page 70), Print Element Syntax (see page 154), and Print Member Syntax (see page 160)— Updated to include the C1PRTVB ddname for the TO clause.
- Define Shipment Destination Syntax (see page 274)— Updated to add the clauses USS Host Path Name Prefix and USS Remote Name Prefix, which enable you to use batch administration to define shipment destinations for USS supported files.
- The Define Shipment USS Mapping Rule Action (see page 282) and Define Shipment USS Mapping Rule Syntax (see page 282)— Added to document this action, which enables you to use batch administration to define mapping rules for USS supported files.
- The Delete Shipment Destination Action (see page 311)— Updated to specify that USS mapping rules are also deleted when a destination is deleted.
- The Delete Shipment USS Mapping Rule Action (see page 312) and Delete Shipment USS Mapping Rule Syntax (see page 312)— Added to enable you to delete USS mapping rules.
- Deleted obsolete notes from the following topics. The notes stated that batch administration could not be used to define package ship destinations or mapping rules for USS supported files.
  - The Build SCL for Shipment Destination Action
  - The Define Shipment Destination Action (see page 273)
  - The Define Shipment Mapping Rule Action (see page 273)
- Define Type Syntax (see page 291)— Updated to add the Element RECFM Is clause that is used by the CA Endevor Quick Edit option to specify whether the ISPF edit data set's record length format is fixed or variable.
- The following changes were made for enhancements to the Autogen option:
  - Set Option Syntax (see page 57)— Updated to add the Autogen Span options.
  - Actions and the Set Options Statement (see page 66)— Updated to add notes about the Autogen option.

- Clear Option Syntax (see page 81)— Updated to state that the Clear Option Autogen statement also clears any Autogen Span options previously set.
- Add Syntax (see page 88)— Updated to add the Autogen Span options. Also added the AUTOGEN_SOURCE option for ENCOPTBL, which changes the behavior of Autogen.
- Generate Syntax (see page 116)— Updated to add the Autogen Span options. Also added the AUTOGEN_SOURCE option for ENCOPTBL, which changes the behavior of Autogen.
- Update Syntax (see page 202)— Updated to add the Autogen Span options. Also added the AUTOGEN_SOURCE option for ENCOPTBL, which changes the behavior of Autogen.

- The Validate Statement— Updated to remove an obsolete statement about there being "no component support for long names."

Release 15.1

- Archive Syntax (see page 104)—Updated to change the minimum LRECL for the Archive statement To File clause.
- How to Allocate Data Sets for Large Elements (see page 170)—Added using text copied from Extended Processors Guide. This information is valid for both the Restore statement and the Transfer statement.
- Transfer from CA Endevor SCM to Archive Data Set Syntax (see page 189)—Updated to clarify a note regarding LRECL for the To File clause.
- Added a note to the following topics, which states that batch administration cannot be used to define package ship destinations or mapping rules for USS supported files:
  - The Build SCL for Shipment Destination Action
  - The Define Shipment Destination Action (see page 273)
  - The Define Shipment Mapping Rule Action (see page 273)

Version 15.0

- Set From Syntax (see page 55)—Updated to add the Retain Generate History option and remove MEMBER removed from syntax diagram.
- Set Options Syntax (see page 57)—Updated to add the Retain Generate History option. Updated to indicate that the NoSource option can be specified as NOSOurce.
- Actions and the Set Options Statement (see page 66)—Updated to add the Retain Generate History option.
- Clear Options Syntax (see page 81)—Updated to add the Retain Generate History option. Updated to indicate that the NoSource option can be specified as NOSOurce.
- The Archive Statement—Updated Element delete behavior.

- The Delete Statement—Updated Element delete behavior.
- Generate Syntax (see page 116)—Updated to indicate that the NoSource option can be specified as NOSOurce.
- Restore Syntax (see page 163)—Updated to add the Retain Generate History option.
- The Transfer Statement (see page 180)—Updated Element delete behavior.
- Transfer from CA Endevor SCM to CA Endevor SCM Syntax (see page 182)—Updated to remove an erroneous reference to FILE | DDNAME.
- Transfer from CA Endevor SCM to Archive Data Set Statement— Updated the diagram FROM clause to add the missing keyword DDNAME. Updated descriptions for Element delete behavior and the specifications for data sets for the TO clause.
- Transfer From Archive Data Set or Unload Tape to CA Endevor SCM Syntax (see page 194)—Updated the diagram FROM clause to add the missing keywords and variable: FILE | DDNAME *ddname*.
- The Backin Package Action (see page 221)—Updated to add the Element action backin option.
- Backin Package Syntax (see page 221)—Updated to add the STATEMENT NUMBER and ELEMENT clauses for the Element action backin option.
- The Backout Package Action (see page 222)—Updated to add the Element action backout option.
- Backout Package Syntax (see page 222)—Updated to add the STATEMENT NUMBER and ELEMENT clauses for the Element action backout option.
- Delete Package Syntax (see page 230)—Updated to correct diagram for the Where Package Status clause.
- Execute Package Syntax (see page 232)—Updated the diagram to indicate that EXECUTE cannot be abbreviated. Also, corrected the diagram for the Where Package Status clause.
- Submit Package Syntax (see page 235)—Updated to correct diagram for the Where Package Status clause.
- Invoking Edit Commands (see page 247)—Added information on how to use an edit command.
- Define Processor Symbol Syntax (see page 272)—Corrected last line of the syntax diagram.
- Define Shipment Destination Syntax (see page 274)—Updated the syntax diagram and added a description for the option REMOTE IPNAME IPPORT.
- Define Subsystem Syntax (see page 284)—Updated to add the clause Exclude Duplicate Processor Output Check.
- Define System Syntax (see page 285)—Updated to add the optional clause Across Subsystems to the clause Duplicate Processor Output Type Check is Active.

- Define Type Syntax (see page 291)—Updated the Element Delta Format specification to add the Log delta format option.

Contents

Chapter 4: Processing Element Actions 85

Chapter 5: Managing Packages in Batch 213

Chapter 1: Introduction

This section contains the following topics:

The Software Control Language (SCL)

Software Control Language (SCL) is a freeform language, with English-like statements, that allow you to manipulate elements, environment definitions, and packages within CA Endevor SCM. SCL is the language used for the non-interactive (batch) execution of CA Endevor SCM. It is a flexible and powerful tool, saving you time in two ways:

- Using SCL allows you to work with as many (or as few) actions as are required to complete a specific job at a particular time.
- Using SCL eliminates much of the screen navigation that is required to process large numbers of elements in an interactive mode.

Because of its consistent nature, SCL is easy to learn and use. For example, you can establish global settings that can be used over and over. This provides a concise and consistent set of options or location information which can be applied to any number of actions, and you need code this information only once in each job stream. Conversely, you can override any pre- established settings by entering like information in a particular request.

There are many features and benefits to using SCL. The following list emphasizes those aspects of SCL that both facilitate and enhance CA Endevor SCM processing.

- SCL allows you to set up a single list or multiple lists of element actions for further manipulation in CA Endevor SCM.
- SCL allows you to manipulate elements or members singly, on a module-by-module level. SCL also allows you to manipulate several library members or module elements at a time. You can tailor your coding to meet your requirements at any time.

- SCL is extremely flexible. You can establish global settings for element action requests (using a SET statement), and override one or all of your selections on a *local level*; that is, within each individual element action request. In addition:
  - You can define the files you want to manipulate either within the language (for example, using a clause such as SET DSNAME...) or external to the language (for example, using a clause such as SET FILE...).
  - You can delay the specification of actions to be run at a particular time, by using the &&ACTION facility (see the description of &&ACTION in the chapter "Element Action Statements"). This capability allows you to define a list of actions for future use and re-use, so you can code only what you need when you need it.
- SCL allows you to mix CA Endevor SCM locations within the same execution. You can change environment, system, subsystem, or type at any time.
- SCL supports processing in type-sequence order, automatically sorting elements according to the specifications determined by the CA Endevor SCM administrator in your organization.
- SCL supports *list processing*. List processing enables you to:
  - Generate a list, edit it as necessary, and break it up into multiple executions instead of keying individual statements.
  - Generate lists based on different selection criteria.
  - Perform configuration management through the use of a special WHERE COMPONENTS EQUAL option.
  - Support a single scan facility that will run against CA Panvalet, CA Librarian, a PDS, and CA Endevor SCM, so you do not need to use separate utilities to scan source code.
- SCL serves as a problem-solving tool, by allowing you to quickly isolate system errors. For example, you can use the WHERE GENERATE FAILED option to generate a list of only those elements that were not successfully processed at a specific time.
- SCL supports vendor interfaces. You can execute SCL from a user-written program, which allows you to write user-defined front-ends for use with various proprietary or vendor-supplied programs.
- SCL enables you to integrate CA Endevor SCM into existing change management/change administration job scheduling systems.
- SCL supports release scheduling (job management). For example, moving a group of elements from a test environment to a production environment any given day.

How Type Sequence Processing Works

A type processing sequence is used when multiple Element types are processed within a single batch request.

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

By default, CA Endevor SCM lets you to define the relative sequence of processing for the various Element types defined within a system. If your site does not choose the Global Type Sequencing option, Element actions execute within your system in the same order as the types were defined to the system and stage by the administrator or as reordered by the administrator using the Type Sequence panel.

When Global Type Sequencing is enabled, Element actions are processed by type sequence regardless of the inventory location of each action. SCL and API Element actions are executed in type sequence order defined at the site level in the Type Sequence member created by the administrator. When Global Type Sequencing is in effect, any types that are not included in the Type Sequence member are processed after all the types in the Type Sequence member have completed processing. Then they are processed in type name order.

Note: For more information about defining the type processing sequence or enabling Global Type Sequencing, see the scenario "How to Enable Global Type Sequencing" in the *Scenario Guide*.

Whether you are using Global Type Sequencing or not, Element action SCL statements are processed in type sequence order. The type specified in the FROM or TO clause determines the sequence in which Element action requests are processed.

An Element's type is indicated in the FROM clause or TO clause (or both). The exact type entry used to determine the processing sequence (that is, type as defined in the FROM clause or the TO clause) depends upon the Element action requested.

The following table summarizes, for each Element action, how the type sequence is determined.

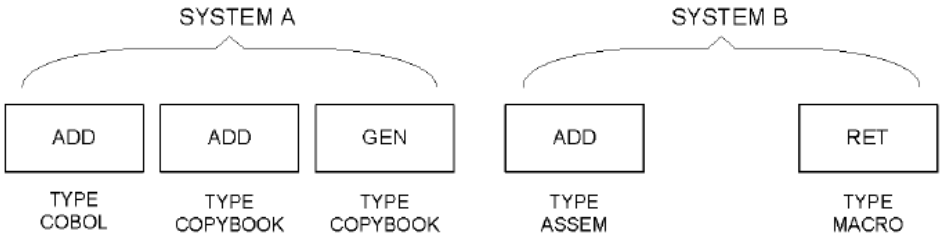
Element Action	Determines Type Sequence
Add	TO clause
Alter	FROM clause
Archive	FROM clause
Delete	FROM clause

Element Action	Determines Type Sequence
Generate	FROM clause
List	FROM clause
Move	FROM clause
Print	FROM clause
Retrieve	FROM clause
Signin	FROM clause
Transfer	TO clause
Update	TO clause

Example: Not Using Global Type Sequencing

Actions are put into the appropriate sequence and executed within each system. In the following illustration, actions have been requested for two systems: System A and System B. Assume that the system administrator has established the following type processing sequences:

- For System A: COPYBOOK, then COBOL
- For System B: MACRO, then ASSEMBLER



As previously illustrated, given the Element type definitions shown for each action, processing would occur in the following sequence.

1. SYSTEM A: ADD ELEMENTS...TYPE COPYBOOK
2. SYSTEM A: GENERATE ELEMENTS...TYPE COPYBOOK...
3. SYSTEM A: ADD ELEMENTS...TYPE COBOL...
4. SYSTEM B: RETRIEVE ELEMENTS...TYPE MACRO
5. SYSTEM B: ADD ELEMENTS...TYPE ASSEM...

Example: Using Global Type Sequencing

Actions are put into the appropriate type sequence and executed as defined by your administrator. For example, assume that the system administrator has established the following type sequence:

1. COPYBOOK
2. MACRO
3. COBOL
4. ASSEMBLER

Given the Element type sequencing definition, processing would occur in the following sequence, regardless of the inventory location.

1. ADD ELEMENTS...TYPE COPYBOOK
2. GENERATE ELEMENTS...TYPE MACRO
3. RETRIEVE ELEMENTS...TYPE MACRO
4. ADD ELEMENTS...TYPE COBOL
5. ADD ELEMENTS...TYPE ASSEMBLER

Process Flow

When you submit your SCL requests, CA Endevor SCM follows a specific processing flow to execute the actions.

Process Flow: Global Type Sequencing Not Enabled

When Global Type Sequencing is not enabled at your site, the process flow is as follows:

1. CA Endevor SCM first parses, or validates, the SCL syntax, assigning a statement number to each SCL statement coded.

   A Syntax Report is produced, echoing the SCL statements entered and flagging any syntax errors.

2. When all requests have been validated, CA Endevor SCM checks for errors. If errors exist within the syntax, processing is terminated.

   If no errors exist, processing continues.

3. CA Endevor SCM checks whether any statements have been entered with an archive file designated as the FROM location. All such actions are performed first, as they are encountered.

   For example, assume you code both an ARCHIVE action and a RESTORE action. If you want CA Endevor SCM to perform the RESTORE action before the ARCHIVE action, designate an archive file as the RESTORE action's FROM location. If you want to perform the ARCHIVE action before the RESTORE action, however, you need to execute SCL twice-first to perform the ARCHIVE action and then to perform the RESTORE action.

   For Elements that are restored, transferred, copied, or listed *from an archive file*, processing occurs as follows:

   a. The Element(s) is restored (or transferred, copied, or listed), but it is not generated at this time.

   b. CA Endevor SCM continues processing the remaining actions, as described in the following steps (beginning with Step 4).

4. CA Endevor SCM expands any name-mask that may have been entered for system, subsystem, stage, and type.

   - Beginning with the first SCL syntax request, CA Endevor SCM checks for use of the name-mask with the system name.

     If a name-mask has not been used with the system name in the first SCL syntax request, CA Endevor SCM checks for the name-mask in the next syntax request. If no name-mask is found and the system name is the same, CA Endevor SCM checks the system name of the third syntax request. This procedure continues until a system name is found with a name-mask or a new system name is encountered, or until all syntax requests have been searched.

     When one of the three situations mentioned above occurs, CA Endevor SCM returns to the first syntax request and checks for a name-mask with the type name.

     Again, if no name-mask is found, the second syntax request is checked, and so on until a type name is found with a name-mask or a new type name is encountered, or until all syntax requests have been checked. This procedure is repeated for stage and subsystem.

     CA Endevor SCM examines each clause (SYSTEM and STAGE) in the syntax request until a non-match is found. Once a difference is encountered, CA Endevor SCM executes the previous syntax requests-.in type sequence order (see Step 5). Processing then continues accordingly with the next syntax request.

   - If a name-mask has been used with the system name in the first syntax request, CA Endevor SCM expands the entries. Then, within each system of the first syntax request, any remaining name-masks are expanded (in the appropriate order).

5. CA Endevor SCM sorts the types based on type sequence order.

   Processing involves syntax requests for stage within a particular system. Type processing sequence conventions still apply, however. If a name-mask is not used with type, the syntax requests themselves are sorted in type sequence order.

   If a name-mask is used with type, actions across all syntax requests are executed in type sequence order. So, depending on the Elements indicated (see Step 6 below), it is not unusual to see an ADD from syntax #2, followed by a GENERATE from syntax #3, followed by an ADD from syntax #2. When all information has been generated for the first (set of matching) syntax request(s), CA Endevor SCM executes the next (set of) syntax request(s).

6. Once all types have been defined, CA Endevor SCM checks the stage identifier involved within the first type. If a name-mask has been used with the stage identifier, CA Endevor SCM expands the entries.

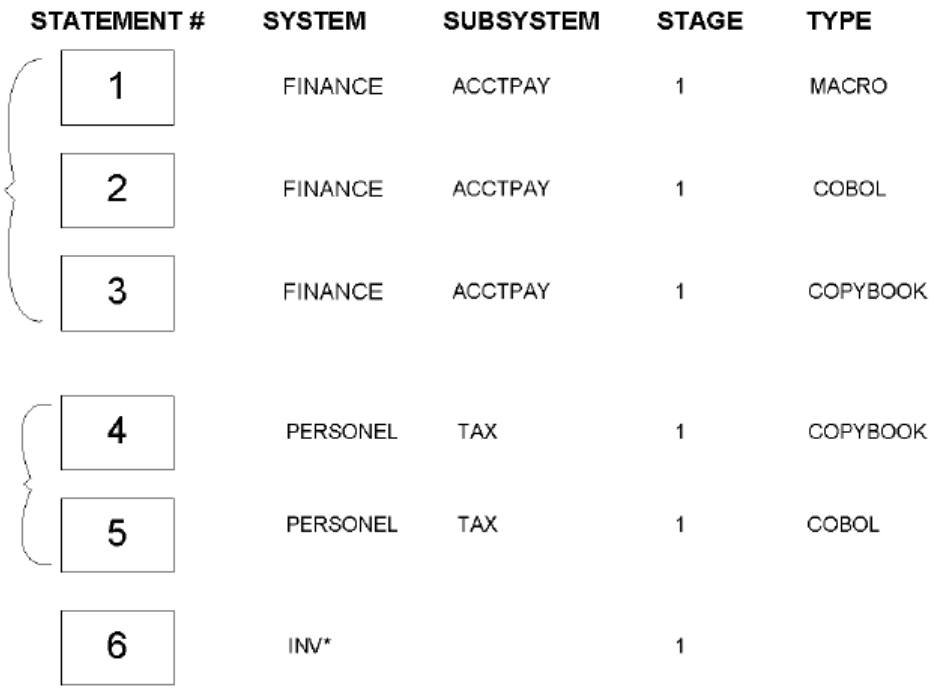
   Still within the first type, and within the first stage identified, CA Endevor SCM expands any subsystem name-masks that have been coded.

7. CA Endevor SCM expands the Element name-mask if it exists (Element is the *Element-name* entered in the first [action] clause of the statement) and executes each action within the system, including those actions previously performed but not generated (because they were from an archive file). Remember: all SCL statements are executed in type sequence order.

8. CA Endevor SCM assigns each action an action number. As all actions are processed, an Execution Report is produced. The Execution Report fully expands the action request, providing the complete system name, subsystem name, type, and stage for the Element being processed. In addition, the report lists all options in effect for the action. CA Endevor SCM also produces a Summary Report. This report provides one line of summary information for each action performed.

Type Processing Sequence Example

This example shows how type processing works when Global Type Sequencing is **not** used. The following illustration displays a typical set of SCL requests:



The type processing sequence has been determined as COPYBOOK, COBOL, MACRO. Processing takes place as follows:

1. CA Endevor SCM first checks the system specification. No name-mask is found, but the system in request #4-PERSONEL-differs from the system in the first three requests-FINANCE.
2. CA Endevor SCM returns to the first request to check the type specification.

   Note: All actions within a particular system are executed at the same time. When a different system name or use of the name-mask is encountered, CA Endevor SCM returns to the first request in the "initial" system and continues processing from that point.

3. The type is different for all three requests in system FINANCE. Because type sequence processing conventions apply, the requests are executed in the following order:

   ```
   Statement #3

   Statement #2
   Statement #1
   ```

   Before the requests are executed, CA Endevor SCM checks whether name-masks have been used with stage. The field is the same for all three requests. Therefore, the actions are executed in the appropriate order.

4. CA Endevor SCM returns to request #4 and checks the system specification in the remaining requests. Both request #4 and request #5 contain the same system-PERSONEL. Request #6, however, contains a different system.

5. CA Endevor SCM returns to request #4 and checks the type specification. Again, the types are different, and the requests are executed in type sequence order:

   ```
   Statement #4
   Statement #5
   ```

   CA Endevor SCM checks the stage and subsystem specifications for a name-mask; none is found. Consequently, CA Endevor SCM continues processing by executing the requests in the order shown above.

6. Request #6 is the last request, and contains a name-mask in the system specification. CA Endevor SCM processes this request by expanding all name-masks encountered in the system, type, stage, and subsystem names and, finally, executing the actions.

Process Flow Using Global Type Sequencing

Process Flow: Global Type Sequencing Enabled

When Global Type Sequencing is enabled at your site, SCL and API Element actions execute in type sequence order defined at the site level by the CA Endevor SCM administrator, regardless of the action's inventory location. Any types that are not included in the Type Sequence member are processed after all the types in the Type Sequence member have completed processing.

All actions are processed in the following order:

1. All actions that have Archive file as the source location. These are executed first, in the Element order found on the archive file. These actions are: Copy, List from Archive, Restore, Transfer from Archive. The execution of these Elements is based on the Element's placement within the archive file. The order of their execution is not affected by the Global Type Sequence option.
2. Non-CA Endevor SCM actions. These include List Member, Print Member, and List ACMQ, which are executed next, in their API/SCL action order. A List ACMQ request is really a list Element request with the following attributes and restrictions:
   - The through Element name cannot be used
   - The CCID, date, or processor where clauses cannot be used
   - Restricted to ACM input or related input component clauses
   - The through component name option cannot be used
   - Build with components and build level none must be specified
3. All other CA Endevor SCM actions. These are executed in type sequence order followed by the statement SCL order. The order of the type records in the site's defined Type Sequence member determines the type sequence processing order.

   The following table lists the actions with the CA Endevor SCM inventory side of each action where inventory expansion and type sequencing is done.

Action	Inventory Location (where inventory expansion and type sequencing occurs)
Add/Update	Target
Alter	Source
Generate	Source
Retrieve	Source
Delete	Source
Move	Source

Action	Inventory Location (where inventory expansion and type sequencing occurs)
Archive	Source
Transfer to Archive	Source
Transfer C1-C1	Target
Signin	Source
Print C1	Source
List C1	Source

Note: When inventory errors are found during the API/SCL expansion phase, these actions are reported as they occur. They always appear before any dispatched CA Endevor SCM action. When a not found condition occurs for an Element or member name with non-explicit inventory, such errors are reported last, after all other actions have been processed.

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 easier to find information and process requests.

Wildcards

A wildcard is an asterisk (*) character that can be used in a search string to represent the entire search string or the end of a search string. It represents any number of characters. When a wildcard is used as the only character of a search string, all members of the search field are returned. When a wildcard is used as the last character of the search string, the only members of the search field returned are those that begin with the characters in the search string preceding the wildcard.

You cannot have more than one wildcard in a string. For example, the statement ADD ELEMENT U*PD* would result in an error.

Examples: Use a Wildcard

- This example shows how to use a wildcard as the only character in a search string. This command adds all Elements.

  ```
  ADD ELEMENT *
  ```

- This example shows how to use a wildcard as the last character of a search string. This command adds all Elements beginning with UPD such as UPDATED or UPDATE.

  ```
  ADD ELEMENT UPD*
  ```

- This example shows how to use the asterisk wildcard as the last character of a search string to select all package IDs beginning with PKG, such as PKGS, PKGB, PKGC, PKGB2, and PKGC5A.

  ```
  PKG*
  ```

Placeholders

A placeholder is a percent sign (%) character that can be used to represent one character in a search string. It can be used at the end of a search string, multiple times within a search string, or both. When a placeholder is used as the last character in a string, all members of the search field are returned, beginning with the characters in the search string preceding the placeholder, but which have no more characters than were used in the search string.

Examples: Use a Placeholder

- This example shows how to use a placeholder to add all Elements with four-character names beginning with UPD such as UPD1 or UPDA.

  ```
  ADD ELEMENT UPD%
  ```

- This example shows how to use the placeholder to return all four-character package IDs beginning with PKG, such as PKGS, PKGB, and PKGC.

  ```
  PKG%
  ```

- This example shows how to use the placeholder multiple times in a single search string to return all Elements with five-character names that have the letter U as the first character and PD as the third and fourth character.

  ```
  ADD ELEMENT U%PD%
  ```

Examples: Use a Wildcard and a Placeholder Together

- This example shows how to use both the wildcard and placeholder to add Elements with names of any length that have U as the first character, any one character as the second character, and D as the third character.

  ```
  ADD ELEMENT U%D*
  ```

- This example shows how to use both the wildcard and placeholder to select all package IDs that have P as the first character, any one character as the second character, and G as the third character, for example, PKGABCD, POGS, PIGGY, PPG1234NDVR, and so on.

  ```
  P%G*
  ```

Valid Uses for Name-Masks

You can use name-masks for the following items:

- On API list requests, name-masking is valid on inventory locations (environment, system, subsystem, type name, and stage ID)
- On CSV utility SCL requests, name-masking is valid on inventory locations (environment, system, subsystem, type name, and stage ID)
- Element names. However, Element name-masks are **not** valid under the following conditions:
  - When entering a LEVel in a statement
  - When using the MEMber clause with an action statement
  - When building a package

- Environment name-masks are valid on the following:
  - On API list requests
  - On CSV utility SCL requests
  - On requests for historical reports
  - On SCL statements, only as follows:
    - On the From Environment field of a Where Components clause
    - On the Alter action From Environment clause, if Search is specified on the Options clause. Alter is the only Element action that supports name-masking the Environment.
    - On the From Environment clause of the following statements: Build SCL For Approver Group, Build SCL For Approver Relations, Build Scl For Environment, Build SCL For Processor Group, Build SCL For Processor Symbol, Build SCL For Subsystem, Build SCL For System, Build SCL For Type, and Build SCL For Type Sequence.
  - On ISPF panels when listing Elements, except that environment name-masks are **not** valid under the following circumstances:
    - On Add/Update panels
    - On Generate panels, the option Copyback cannot be specified when the Environment name is masked
    - On any panels where environment name-masking is allowed, the following list options are ignored when you mask the environment name: Build Using Map, Return First Found, Display Sys/sbs List
- ISPF panels
- System, subsystem, and type names within From clauses
- Report syntax, except environment name-masking is only allowed on historical reports
- Package IDs
- Parallel Development Option (PDM) does support name-masks on CA Endevor SCM inventory locations (environment, system, subsystem, type, and stage)

SCL Statement Syntax Conventions

The SCL statement syntax presented in the CA Endevor SCM documentation uses the following notation conventions.

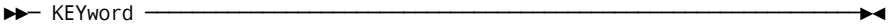
This syntax represents the beginning of a syntax statement:

This syntax represents the end of a syntax statement:

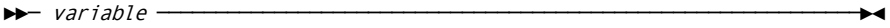
This syntax represents the continuation of a syntax statement to the following line:

This syntax represents the continuation of a syntax statement from the preceding line:

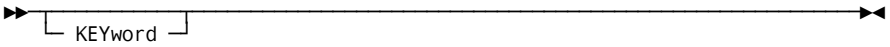
This syntax represents a required keyword. Only the uppercase letters are necessary:



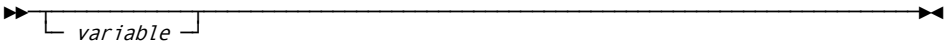
This syntax represents a required user-defined variable:



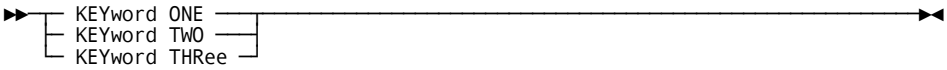
This syntax represents an optional keyword. Optional keywords appear below the syntax line. If coded, only the uppercase letters are necessary:



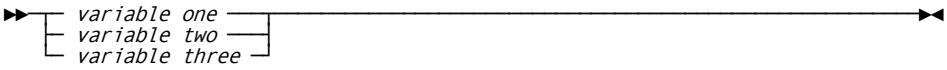
This syntax represents an optional user-defined variable. Optional variables appear below the syntax line:



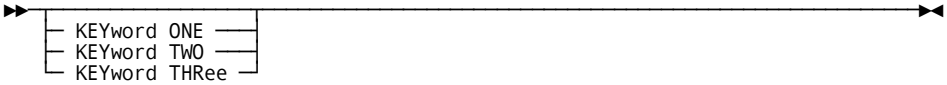
This syntax represents a choice of required, mutually exclusive keywords. You must choose only one keyword:



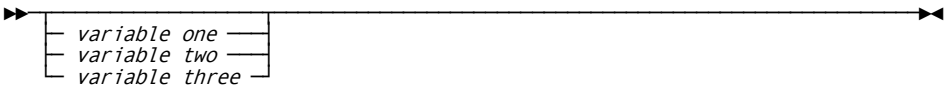
This syntax represents a choice of required, mutually exclusive, user-defined variables. You must choose only one variable:



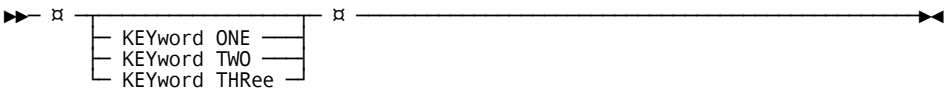
This syntax represents a choice of optional, mutually exclusive keywords. Optional keywords appear below the syntax line:



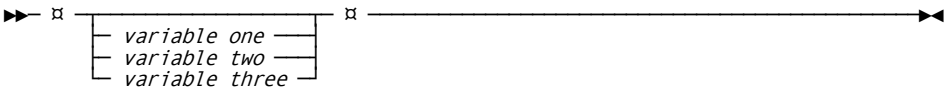
This syntax represents a choice of optional, mutually exclusive, user-defined variables. Optional variables appear below the syntax line:



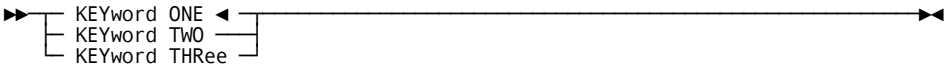
This syntax represents a choice of optional keywords. The boxed-shaped stars (¤) indicate that the keywords are not mutually exclusive. Do not code any keyword more than once:



This syntax represents a choice of optional user-defined variables. The boxed-shaped stars (¤) indicate that the variables are not mutually exclusive. Do not code any variable more than once:

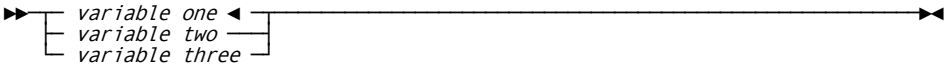


This syntax represents a choice of required, mutually exclusive keywords, one of which is the default. In this example, KEYword ONE is the default keyword, because it is shown on the line with an arrow pointing to it:

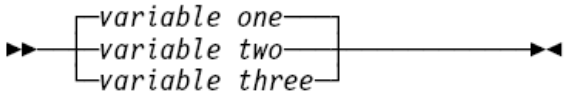


Note: A default can be shown one of two ways in the syntax diagrams. The default is shown either above the line or on the line with an arrow pointing to it. In this example, KEYword ONE is the default keyword, because it is shown above the line:

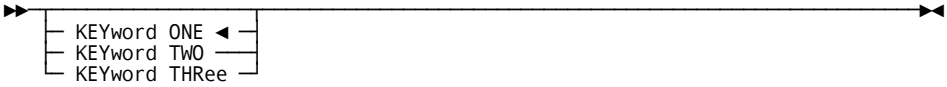
This syntax represents a choice of required, mutually exclusive, user-defined variables, one of which is the default.In this example, variable one is the default keyword, because it is shown on the line with an arrow pointing to it:



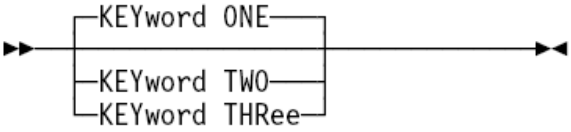
Note: A default can be shown one of two ways in the syntax diagrams. The default is shown either above the line or on the line with an arrow pointing to it. In this example, variable one is the default variable because it appears above the syntax line:



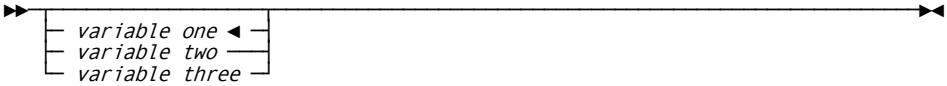
This syntax represents a choice of optional, mutually exclusive keywords, one of which is the default. In this example, KEYword ONE is the default keyword, because it is shown on the line with an arrow pointing to it:



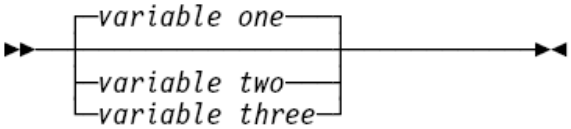
Note: A default can be shown one of two ways in the syntax diagrams. The default is shown either above the line or on the line with an arrow pointing to it.. In this example, KEYword ONE is the default keyword because it appears above the syntax line:



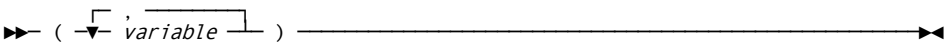
This syntax represents a choice of optional, mutually exclusive, user-defined variables, one of which is the default. In this example, variable one is the default keyword, because it is shown on the line with an arrow pointing to it:



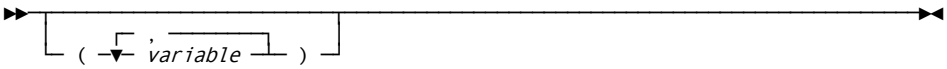
Note: A default can be shown one of two ways in the syntax diagrams. The default is shown either above the line or on the line with an arrow pointing to it. In this example, variable one is the default variable because it appears above the syntax line:



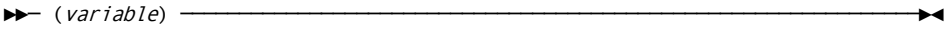
This syntax represents a required variable that can be repeated. Separate each occurrence with a comma and enclose all variables in a single set of parenthesis:



This syntax represents an optional variable that can be repeated. Separate each occurrence with a comma and enclose all variables in a single set of parenthesis:

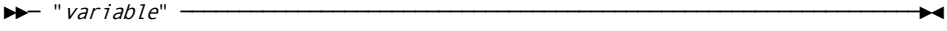


This syntax represents a variable which must be enclosed by parenthesis:

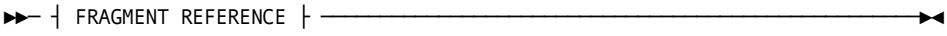


This syntax represents a variable which must be enclosed by single quotes:

This syntax represents a variable which must be enclosed by double quotes:

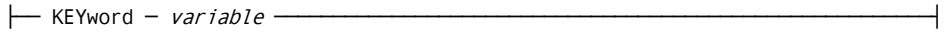


This syntax represents a reference to a syntax fragment. Fragments are listed on the lines immediately following the required period at the end of each syntax statement:

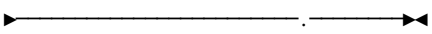


This syntax represents a syntax fragment:

Expansion of FRAGMENT

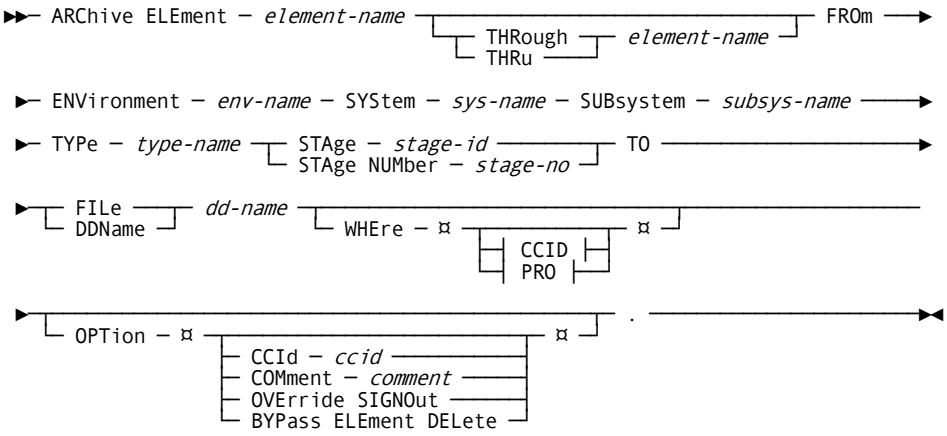


This syntax represents the period required at the end of all syntax statements:

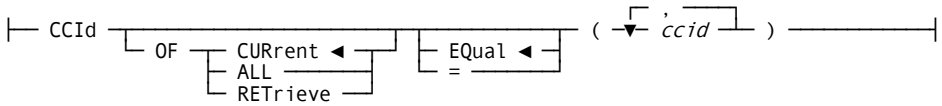


Syntax Diagram

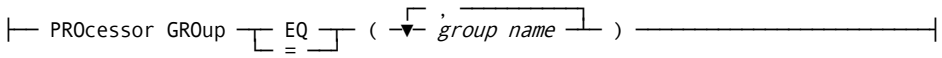
The CA Endevor SCM syntax diagram lets you quickly see how to construct a statement or command. The following is a sample syntax diagram for the Archive Element action.



Expansion of CCID



Expansion of PRO



This sample syntax diagram contains the following syntax:

ARChive ELEment *element-name*

The keyword ARChive ELEment appears on the main line, indicating that it is required. The variable element-name, also on the main line, must be coded.

THRough / THRu *element-name*

The keywords THRough and THRu appear below the main line, indicating that they are optional. They are also mutually exclusive.

FROm ENVironment ... TYPe *type-name*

Each keyword and variable in this segment appears on the main line, indicating that they are required.

STAge *stage-id* / STAge NUMber *stage-no*

The keywords STAge and STAge NUMber appear on and below the main line, indicating that they are required, mutually exclusive keywords.

TO ... *dd-name*

The keyword TO appears on the main line, indicating that it is required. The keywords FILe and DDName appear on and below the main line, indicating that they are required, mutually exclusive keywords. The variable dd-name also appears on the main line, indicating that it is required.

WHERe *clause*

This clause appears below the main line, indicating that it is optional. The keyword WHERe appears on the main line of the clause, indicating that it is required. CCID and PRO are syntax fragments that appear below the main line, indicating that they are optional. The boxed-shaped stars (¤) indicate that they are not mutually exclusive.

OPTion *clause*

This clause appears below the main line, indicating that it is optional. The keyword OPTion appears on the main line of the clause, indicating that it is required. The keywords CCId, COMment, OVErride SIGNOut, and BYPass ELEment DELete all appear below the main line, indicating that they are optional. The boxed-shaped stars (¤) indicate that they are not mutually exclusive.

CCID *fragment*

The keyword CCId appears on the main line, indicating that it is required. The OF clause appears below the main line, indicating that it is optional. If you code this clause, you must code the keyword OF, as it appears on the main line of the clause. CURrent, ALL, and RETrieve appear above, on, and below the main line of the clause, indicating that they are required, mutually exclusive keywords. CURrent appears above the main line, indicating that it is the default. If you code the keyword OF, you must choose one and only one of the keywords.

The keywords EQual and = appear above and below the main line, indicating that they are optional, mutually exclusive keywords. EQual appears above the main line, indicating that it is the default. You can include only one. The variable ccid appears on the main line, indicating that it is required. The arrow indicates that you can repeat this variable, separating each instance with a comma. Enclose all variables in a single set of parenthesis.

PRO *fragment*

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

Rules for Coding Syntax

When coding syntax, you must follow certain rules and guidelines regarding valid characters, incompatible commands and clauses, and ending statements. In addition, knowing how the SCL parser processes syntax will help you resolve errors and undesired results. The following information outlines the rules and guidelines you should follow when coding syntax.

Valid Characters (Allowed when Coding Syntax)

- Uppercase and lowercase letters A through Z
- Numbers 0 through 9
- National characters:
  - The at sign @
  - Dollar sign $
  - Pound sign #

  Note: The system recognizes the following hexadecimal representations of the U.S. National characters: @ as X'7C'; $ as X'5B'; and # as X'7B'. In countries other than the U.S., the U.S. National characters represented on terminal keyboards might generate a different hexidecimal representation and cause an error. For example, in some countries the $ character may generate a X'4A'.
- Hyphens (-)
- Underscores (_)

Valid Characters (Must be enclosed in single (') or double (“) quotes)

- Spaces
- Tabs
- New line
- Carriage return
- Comma (,)
- Period (.)
- Equal sign (=)
- Greater than sign (>)
- Less than sign (<)
- Parenthesis ( )

- Single quotation marks—A string containing single quotation marks must be enclosed in double quotation marks.
- Double quotation marks—A string containing double quotation marks must be enclosed in single quotation marks.

Note: Null (X'00') characters are translated to spaces (X'40') in the text of Comments and CCIDs.

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

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

Important! The characters "*" and "%" are reserved for name-masking.

Incompatible Commands and Clauses (mutually exclusive)

- THRough and MEMber clauses within any action except LIST
- CA Endevor SCM location information (environment, system, subsystem, type, and stage) and data set names (DSName)
- File names (DDName) and data set names (DSName)
- The stage id (STAge / STAge ID) and the stage number (STAge NUMber)
- The SET TO CA Endevor SCM location information and the SET TO MEMber clause

Syntax to End a Statement

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

&&ACTION Statement

If you use the &&ACTION statement, you must have previously coded a SET ACTION statement. Refer to the descriptions of SET ACTION in “Set, Clear, and EOF Statements”, and the description of &&ACTION in “Element Action Statements” for complete coding information.

SCL Parsing Information

- The SCL parser does not look for information in columns 73-80 of the input. Therefore, be sure that all relevant information is coded in columns 1-72.
- If columns 1-72 of the input are blank, the line is ignored.
- The SCL parser does not catch duplicate clauses coded for an SCL request. If you code the same clause more than once, the last entry is used.

- Rules for valid comments follow:
  - If you enter an asterisk (*) in column 1, the remainder of the line is considered a comment by the SCL parser and is ignored during processing.
  - Any value found to the right of the period terminating the SCL statement is considered a comment by the SCL parser and is ignored during processing. A blank space must separate the period from the comment.
  - Columns 73-80 are ignored and can be used as a comment area.

Example: Duplicate keywords in SCL Statement

Duplicate keywords do not make SCL statements invalid. In the case of duplicate keywords, the last keyword is used. The following SCL statement is valid. Stage 1 and Type ASMPGM are ignored, and Stage 2 and Type COBOL are used.

```
 GENERATE ELEMENT XZY FROM ENV 'ENV1' SYSTEM 'ESCM140' SUBSYSTEM 'GA'   3RD STMT
STAGE 1
 STAGE 2
  TYPE ASMPGM
  TYPE COBOL
  .
```

Examples: Comments in SCL Statements

The following examples show valid and invalid comments in SCL statements:

- The comment on the last line is valid, because it is to the right of the period and is separated from the period by a blank space. The comments on the first two lines are valid because they are preceded by an asterisk in the first column of each line.

```
** Valid: The comment on the last line is valid because it is to the right of
** the period and is separated from the period by a blank space.
GENERATE ELEMENT XZY FROM ENV 'ENV1' SYSTEM 'ESCM140' SUBSYSTEM 'GA'
TYPE ASMPGM STAGE A
 . This is a valid comment on an SCL statement.
```

- The comment on the last line is **not** valid, because, even thought it is to the right of the period, there is no blank space after the period.

```
 GENERATE ELEMENT XZY FROM ENV 'ENV1' SYSTEM 'ESCM140' SUBSYSTEM 'GA'
  TYPE ASMPGM STAGE NUM 1
  ., This is an invalid comment on an SCL statement.
```

- The comment on the last line is **not** valid, because there is no period to separate SCL the from the comment. This error message would result: E004 INVALID COMMAND WORDING, FOUND: this. The parser catches the word this. If you add a blank between ASMPGM and /*, /* would be flagged as the invalid command word.

```
 GENERATE ELEMENT XZY FROM ENV 'ENV1' SYSTEM 'ESCM140' SUBSYSTEM 'GA'
STAGE 1
  TYPE ASMPGM/* this is another invalid comment */
  .
```

SCL Continuation Syntax Rules

SCL keyword parameters cannot span multiple lines; however, the parameter values can span multiple lines. All SCL parameter values that span multiple lines must be enclosed in single or double quotes. The syntax required to span a parameter value must start with a single or double quote at the beginning of the specification and a trailing single or double quote at the end of the value. Spaces at the beginning or end of the spanned lines must be surrounded by non-blank characters in order to be included in the text string. Example:

```
ADD ELEMENT 'Spanned
 ElementName' CCID 'CC
 ID001'
```

This would result in an element value of "SpannedElementName" and a CCID value of "CCID001".

Long File and Path Name Syntax

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

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

HFSFile Syntax Rules

A filename can be up to 255 characters long. To be portable, the filename should use only the characters in the POSIX portable filename character set:

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

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

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

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

A filename can include a suffix, or extension, that indicates its file type. An extension consists of a period (.) and several characters. For example, files that are C code could have the extension **.c**, as in the filename **dbmod3.c**. Having groups of files with identical suffixes makes it easier to run commands against many files at once.

Path Name Syntax Rules

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

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

Element Name Syntax Rules

An element name can be up to 255 characters long. However, element names longer than ten characters, or in mixed or lower case, cannot be accessed in foreground.

Element names can contain only the following characters:

- Uppercase letters
- Lowercase letters
- Numbers
- National characters
- Period (.)
- Hyphen (-)
- Underscore(_)
- Left bracket ({) character, for all except the first character

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

Note: IBM allows the left bracket ({) character in the member name of a PDS, but not as the first character. However, IBM does *not* allow the left bracket character in a data set name.

Chapter 2: Using SCL

This section contains the following topics:

SCL Language Overview

The SCL language consists of SCL statements written in an easy-to-follow format. The format must always include an action, an element, and one or more required clauses. Optional clauses can be added to the request to provide CA Endevor SCM with additional information about the selected elements. This section discusses general coding information as well as coding conventions unique to CA Endevor SCM and SCL.

SCL Statements

There are several types of SCL statements:

- Set statements
- Clear statements
- EOF (EOJ) statement
- Element action statements (also referred to as action statements)
- Environment definition statements
- Package action statements

Note: SET and CLEAR statements apply only to element action statements.

Set Statements

SET statements are global default statements that establish values for subsequent element action statements. A SET statement establishes applicable keyword values (for example, FROM and TO) for specific items that may be omitted from selected action statements. If a certain parameter is used (or required) but not coded in a particular action statement, CA Endevor SCM looks for that information in a corresponding SET statement.

SET statements also allow consistency across several actions. If you want to use a particular option (such as CCID or comments) for several actions or perform actions against those elements in a specific location (TO or FROM), code the appropriate SET statement. The data you enter is applied to every subsequent action. SET statements are in effect until another SET statement or a CLEAR statement is encountered or processing ends.

You can define the following SET statements:

- SET ACTION
- SET BUILD
- SET FROM
- SET OPTIONS
- SET STOPRC
- SET TO
- SET WHERE

Clear Statements

A CLEAR statement clears the information designated by a related SET statement. When you are working with a series of element actions and need to remove information established in a SET statement, code a parallel CLEAR statement. The CLEAR statement remains in effect until you enter another related SET statement or until processing ends.

CLEAR statements only apply to SET statements. Similar information entered in an element action statement is not affected by a CLEAR statement.

You can use the following CLEAR statements:

- CLEAR BUILD
- CLEAR FROM
- CLEAR OPTIONS
- CLEAR TO
- CLEAR WHERE

The EOF (EOJ) Statement

The EOF or EOJ (End of File or End of Job) 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 perform.

You can enter either EOF or EOJ. Use the value to which you are most accustomed.

Element Action Statements

Element action statements operate against an Element or a group of Elements. The Element actions consist of the following:

- *ADD*-Puts a member under CA Endevor SCM control from an external data set.
- *ALTER*-Replaces Element record metadata in the Master Control File with user-specified values.
- *ARCHIVE*-Writes the current version of an Element to a sequential file (or archive data set).
- *COPY*-Copies an Element from an archive data set to a data set external to CA Endevor SCM.
- *DELETE*-Erases base and delta forms of an Element and removes related information from a master control file or component list.
- *GENERATE*-Creates an executable form of an Element.
- *LIST*-Creates a list of Elements or members that meet specific selection criteria.
- *MOVE*-Moves Elements between stages, within or across environments.
- *PRINT*-Prints Element or member information.
- *RESTORE*-Restores Elements to CA Endevor SCM from an archive data set.
- *RETRIEVE*-Copies Elements from CA Endevor SCM to an external data set.
- *SIGNIN*-Removes the user signout associated with an Element.
- *TRANSFER*-Transfers an Element from one location to another: CA Endevor SCM to CA Endevor SCM, CA Endevor SCM to an archive data set, or archive data set to CA Endevor SCM.
- *UPDATE*-Updates an Element from an external data set.
- *VALIDATE*-Checks to make sure that Elements were generated correctly, no synchronization errors are detected, and that all the components exist and are valid.

Environment Definition Statements

Environment definition statements provide the ability to manage environment definitions for all inventory structures using SCL. Environment definition statements allow you to create, update, and delete inventory definitions. The statements manage the following environment definitions:

- Approver group
- Approver group relationships
- Element types
- Package shipment data set mapping rules
- Package shipment destination
- Processor groups
- Processor symbols
- Subsystem
- Systems
- Type sequence

Package Action Statements

Package action statements provide the ability to perform package processing in batch using SCL. Package action statements consist of the following:

- APPROVE PACKAGE-Approves a package for execution.
- ARCHIVE PACKAGE-Copies the package definitions to an external data set.
- BACKIN PACKAGE-Backs a package in, reversing the BACKOUT PACKAGE action.
- BACKOUT PACKAGE-Backs out the change package to restore the executable and output modules to the state they were in prior to execution.
- CAST PACKAGE-Casts a package, which freezes the data and prevents further changes at that time.
- COMMIT PACKAGE-Commits a package removing all backout/backin data.
- DEFINE PACKAGE-Creates a new or updates an existing package.
- DELETE PACKAGE-Deletes an entire package from CA Endevor SCM.
- DENY PACKAGE-Denies execution of a package.
- EXECUTE PACKAGE-Executes a package.

- EXPORT PACKAGE-Writes the SCL associated with a package to an external data set.
- INSPECT PACKAGE-Checks each element for security, signout, and synchronization conflicts and source changes and reports on the changes in element status that might effect the successful execution of the package.
- RESET PACKAGE-Resets a package back to a status of In-edit.
- SUBMIT PACKAGE-Submits a JCL job stream to execute one or more packages.

Note: For information about package processing, see the *Packages Guide*.

Statements and Clauses

References are made to *statements* and *clauses* throughout this manual. For SCL purposes, these terms are defined as follows:

- A *statement* begins with an action (for example, ADD or DEFINE) and ends with a period (**.**). A statement consists of one or more clauses, depending on how you code the SCL syntax.
- A *clause* is an individual line of information within each statement (for example, FROM ENVIRONMENT TEST or WHERE CCID EQ 'FIX01'). Any number of clauses may be contained within one statement.

Example: Using Statements and Clauses

In this example, Lines 1-7 form a statement. Line 1 begins with an action (MOVE) and line 7 ends with a period. Lines 2-5 constitute a single clause (a FROM clause). Lines 6 and 7 are individual clauses. Each of these clauses provide information essential to the statement.

```
1.MOVE ELEMENTS COPY01

2.     FROM       ENVIRONMENT        DEMO

3.                SYSTEM             FINANCE

4.                SUBSYSTEM          ACCTPAY

5.                TYPE               COBOL

6.     WHERE CCID EQ          FIX'

7.     OPTIONS                WITH HISTORY.
```

When you use statements and clauses, you must enter the action clause first. You can enter the remaining clauses in any order. Within each clause, however, you must code the sub-clauses in the order in which they are shown in the syntax.

In the previous example, you might code the FROM clause last and the OPTIONS clause immediately after the MOVE ELEMENTS clause. Within the FROM clause, though, you must enter ENVIRONMENT first, followed by SYSTEM, followed by SUBSYSTEM, followed by TYPE.

Element Action Examples

The following examples demonstrate different ways you can use the Element action SCL. The four examples all produce the same result; the only difference is in the number and types of statements and clauses used.

Note: The examples shown here apply to the general structure of environment definition and package action syntax. The major difference, and the reason examples are shown for the Element actions, is the use of SET and CLEAR statements.

Example: Element Action Using Long-hand SCL

In this example, the Element action uses long-hand SCL. The TRANSFER, FROM, TO, WHERE, and OPTIONS statements are repeated for each Element.

```
TRANSFER ELEMENT COPY01
  FROM  ENVIRONMENT                    DEMO
        SYSTEM                         FINANCE
        SUBSYSTEM                      ACCTPAY
        TYPE                           COPYBOOK
        STAGE                          NUMBER2
  TO    ENVIRONMENT                    PROD
        STAGE                          NUMBER1
        WHERE    CCID EQ               'FIX01'
        OPTIONS  COMMENT               'FIX BUG'
.
TRANSFER ELEMENT COPY02
  FROM  ENVIRONMENT                    DEMO
        SYSTEM                         FINANCE
        SUBSYSTEM                      ACCTPAY
        TYPE                           COPYBOOK
        STAGE                          NUMBER2
  TO    ENVIRONMENT                    PROD
        STAGE                          NUMBER1
        WHERE    CCID EQ               'FIX01'
        OPTIONS  COMMENT               'FIX BUG'
.
TRANSFER ELEMENT PROG02
  FROM  ENVIRONMENT                    DEMO
        SYSTEM                         FINANCE
        SUBSYSTEM                      ACCTPAY
        TYPE                           COBOL
        STAGE                          NUMBER2
  TO    ENVIRONMENT                    PROD
        STAGE                          NUMBER1
        WHERE     CCID EQ              'FIX01'
        OPTIONS    COMMENT             'FIX BUG
.
```

Note that the information coded in the FROM clauses (except in the last FROM clause where TYPE is different), TO clause, WHERE clause, and OPTIONS clause is the same. Although there is nothing wrong with coding every line of a request, you may find it time-consuming when you need to code several requests. Therefore, it is important to consider several "shortcuts" when coding the Element action syntax. Examples 2 - 4 demonstrate these shortcuts.

Example: Element Action Using Global Settings

In this example, global settings are used with SET statements to assign the location (FROM and TO) information, as well as common WHERE and OPTIONS data.

```
SET FROM                ENVIRONMENT     DEMO
                        STAGE           NUMBER2.
SET TO                  ENVIRONMENT     PROD
                        STAGE           NUMBER1.
SET WHERE               CCID EQ         'FIX01'
SET OPTIONS             COMMENT         'FIX BUG'.
TRANSFER ELEMENT                        COPY01.
TRANSFER ELEMENT                        COPY02.
SET FROM                TYPE            COBOL.
TRANSFER ELEMENT                        PROG01.
```

In this example, all SET statements coded at the beginning of the syntax are applied to the first two TRANSFER action requests. Because the type is different for the third TRANSFER action request, however, a new SET FROM statement has been entered-containing only the different information.

This new type will be applied to the subsequent TRANSFER request. But, all other previously-coded information will be applied also. Remember: the data entered in a SET statement remains in effect until a new, like SET statement (or a CLEAR statement) is encountered.

Example: Element Action Using a Combination of Global and Local Settings

In this example, a combination of global and local settings are used, and the SET statements are applied to all three TRANSFER action requests, with the exception of type in the third request.

```
SET FROM          ENVIRONMENT    DEMO
                   SYSTEM        FINANCE
                   SUBSYSTEM     ACCTPAY
                   TYPE          COPYBOOK
                   STAGE         NUMBER2.
SET TO            ENVIRONMENT    PROD
                   STAGE         NUMBER1.
SET WHERE          CCID          EQ'FIX01'.
SET OPTIONS        COMMENT       'FIX BUG'.
TRANSFER ELEMENT                 COPY01.
TRANSFER ELEMENT                 COPY02.
TRANSFER ELEMENT                 PROG01
   FROM           TYPE           COBOL.
```

Remember: a value entered locally overrides a like value in a SET statement. Therefore, coding the clause FROM TYPE COBOL is all that is required in the third request. The remaining location, WHERE, and OPTIONS information defaults to the entries coded in the previous SET statements.

Example: Element Action Using a Name-Mask

In this example, a name-mask is used to indicate that all Elements beginning with the indicated letters should be considered for an action.

```
TRANSFER ELEMENT ABC*
    FROM  ENVIRONMENT    DEMO
          SYSTEM         FINANCE
          SUBSYSTEM      ACCTPAY
          TYPE           *
          STAGE          NUMBER2
    TO    ENVIRONMENT    PROD
          STAGE NUMBER   1
    WHERE  CCID EQ       'FIX01'.
    OPTIONS COMMENT      'FIX BUG'.
```

In this example, use of the asterisk alone in the TRANSFER ELEMENTS clause indicates that all Elements-as long as the remaining selection criteria is met-should be selected for the TRANSFER. Use of the name-mask in the TYPE clause indicates that any type will be acceptable in the TRANSFER action.

Using the name-mask with the Element name and the type eliminates the need to set and change SET statements (as was done in examples 2 and 3). Example 4 instructs CA Endevor SCM to look for all Elements, no matter what type, from the CA Endevor SCM location indicated (in the environment, system, subsystem, and stage number clauses), associated with a CCID of FIX01. And, the comment FIX BUG will be applied to all Elements meeting that selection criterion.

Chapter 3: Using Set, Clear and EOF Statements

This section contains the following topics:

Set Statements

A SET statement sets up applicable keyword values (for example, FROM, TO) for specific items that are omitted from subsequent element action statements. If a parameter is required and not specifically coded with an element action statement, a corresponding SET statement must precede that action statement. The SET statement can be reissued to change the default value of a particular keyword any number of times within an SCL stream.

You can remove a SET statement by using a CLEAR statement for the same keyword. Be sure to issue the CLEAR statement after the related element action statement; otherwise, the SET statement is canceled and you may receive an error message. (CLEAR statements are explained later in this chapter.)

Note: The SET statement establishes default values; it is never executed. Therefore, no element processing is involved.

Set Statement Conventions

The following conventions apply to all SET statements.

- SET statements are applied globally to all element action statements following the entry. Each SET statement remains in effect until one of the following conditions occurs:
  - 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. For example, a CLEAR WHERE statement would cancel a SET WHERE statement.
  - Processing for this job ends; that is, an EOF or EOJ statement is encountered.

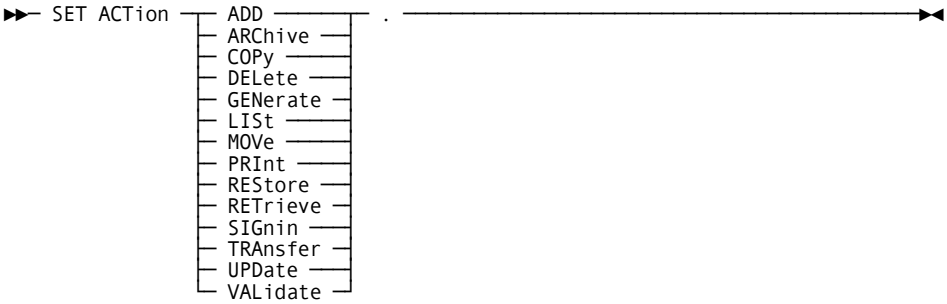
- SET statements, and the information contained in each, apply only where similar data appears on a "local" level; that is, within a specific action statement. For example, if one of the actions following a SET TO statement does not require any TO data, the SET TO statement is ignored.
- Information in the SET statement will be replaced by any overriding SET values coded locally. That is, if the element action syntax contains the variable specified in the SET statement, the like information in the SET statement is ignored. For example, if you enter system and subsystem names in the FROM clause for a COPY action, CA Endevor SCM uses those names rather than the names coded in the related SET FROM statement.
- If the information is not available in the element action statement, the like information in the SET statement is applied to the syntax. For example, if you do not code a system and subsystem in the FROM clause for the COPY action, the information will be taken from the related SET FROM statement.

The Set Action Statement

The SET ACTION statement is used in conjunction with the &&ACTION statement. When you use this statement, CA Endevor SCM sets the action in all following &&ACTION statements to the action indicated. The specified action applies until the system encounters another SET ACTION or a CLEAR ACTION statement, or when processing is terminated.

Set Action Syntax

The following is the syntax for set action:



When you use this statement, CA Endevor SCM sets the action in all following &&ACTION statements to the action you indicate in this statement. The action specified applies until the system encounters another SET ACTION or a CLEAR ACTION statement, or when processing is terminated.

Although you can enter more than one SET ACTION statement in your syntax, only the action indicated in the SET ACTION statement immediately preceding the &&ACTION statement is performed.

You can code the following actions in the SET ACTION statement:

- ADD
- ARCHIVE
- COPY
- DELETE
- GENERATE
- LIST
- MOVE
- PRINT
- RESTORE
- RETRIEVE
- SIGNIN
- TRANSFER
- UPDATE
- VALIDATE

The Set Build Statement

The SET BUILD statement applies only to the BUILD statement in the LIST action (see the explanation of LIST earlier in this chapter). This statement has three parts:

ACTION

Determines the action that is placed in the list of action cards generated by the LIST request.

LEVEL

Determines whether the element current version and level is listed.

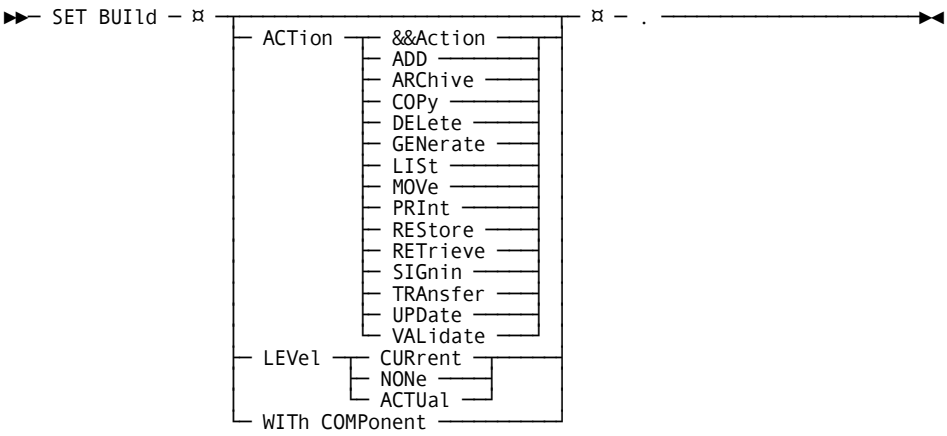
WITH COMPONENTS

Determines whether a component list should be included in the listing for the specified element.

Note: The WITH COMPONENTS option pertains to the CA Endevor SCM ACM product only.

Set Build Syntax

The following is the syntax for set build action:



SET BUILD ACTION applies to any Element, whether from CA Endevor SCM or an external file (that is, a sequential file or a library). The action coded stays in effect until CA Endevor SCM encounters the next SET BUILD ACTION or a CLEAR BUILD ACTION statement, or processing ends.

The following can be coded in the SET BUILD ACTION statement:

&&ACTION

Indicates that an action will be designated for this Element at a later time.

ADD

Adds an Element to the environment's entry stage in CA Endevor SCM.

ARCHIVE

Writes the current version of an Element to a sequential file (or archive data set).

COPY

Copies an Element from an archive data set to a data set external to CA Endevor SCM.

DELETE

Removes an Element from either Stage 1 or Stage 2 in CA Endevor SCM.

GENERATE

Executes the generate processor for the current level of the Element.

LIST

Lists Elements from the Master Control File or an archive data set, or lists members from a library.

MOVE

Moves Elements from one map location to another.

PRINT

Prints either information relating to an Element (if executed against CA Endevor SCM) or the source of the selected members (if executed against an external library).

RESTORE

Restores an Element from an archive data set back to CA Endevor SCM.

RETRIEVE

Copies an Element from either stage to a user data set (a sequential file, library, or PDS).

SIGNIN

Removes the user signout associated with either a Stage 1 or a Stage 2 Element.

TRANSFER

Transfers an Element from one location to another (CA Endevor SCM to CA Endevor SCM, CA Endevor SCM to an archive data set, or archive data set/unload tape to CA Endevor SCM).

UPDATE

Updates an Element in the environment's entry stage only.

VALIDATE

Checks to make sure that Elements were generated correctly, no synchronization errors are detected, and that all the components exist and are valid.

SET BUILD LEVEL

Applies only to Elements in CA Endevor SCM (as opposed to those Elements currently in external files). The level coded stays in effect until CA Endevor SCM encounters the next SET BUILD LEVEL or a CLEAR BUILD LEVEL statement, or processing ends. The following options apply to the SET BUILD LEVEL statement.

CURRENT

If the WHERE COMPONENTS EQUAL clause has not been coded for the action, or no component list exists (that is, the CA Endevor SCM ACM product is not installed), the system defaults to the current level of the Element.

NONE

The current version and level of the Element are not to be listed on the action cards generated by the LIST request.

ACTUAL

The actual level of each component as recorded in the component list, rather than the current level of the Element as recorded in the Master Control File, should be used to build the Element action statement.

If the WHERE COMPONENTS EQUAL clause has not been coded, or no component list exists, (that is, CA Endevor SCM ACM product is not installed), the current level of the Element is listed.

SET BUILD WITH COMPONENTS

Indicates that action cards should be generated for every input component that is associated with the specified Element. If you enter this clause, you must also have a WHERE COMPONENTS EQUAL clause coded, either in the LIST action or as part of a SET WHERE statement. SET BUILD WITH COMPONENTS is in effect until the system encounters a CLEAR BUILD WITH COMPONENTS statement or processing ends.

Note: This option pertains to the CA Endevor SCM ACM product only.

The Set From Statement

The SET FROM statement applies to each element action that uses-but does not contain all or part of-a FROM clause, and remains in effect until the system encounters another SET FROM statement or a CLEAR FROM statement, or when processing ends.

The exact information used from the SET FROM statement depends on both the specific action and the data you have entered in that action statement. What you enter in the action's FROM clause overrides that particular entry in the SET FROM statement. For example, you code all CA Endevor SCM location information (environment, system, subsystem, type, and stage number or stage ID) in a SET FROM statement. Then, when coding a RETRIEVE statement, you enter a different type. CA Endevor SCM determines the FROM location by applying all SET FROM information except for the type, which is taken from the RETRIEVE statement.

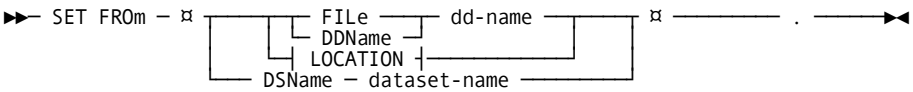
Three types of information can be provided by the SET FROM statement, depending on the action you enter.

- Some actions require only CA Endevor SCM location information.
- Some actions require only a file name (DDname) or data set name.
- Some actions require both a file name (DDname) and CA Endevor SCM location information.

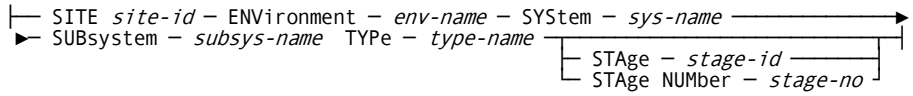
The following section contains information about each type of information. Refer to the individual element action descriptions to determine the requirements for each action.

Set From Syntax

The following is the syntax for the set from action:



Expansion of LOCATION



SET FROM

Specifies default parameters for the From clause. Often, an action requires that only a file name or data set name be entered to indicate a From location. Occasionally, you are required to enter both a file name and CA Endevor SCM location information for the element. In this case you must enter the file name (DDname) before the location information, otherwise that data is ignored and you receive an error message. You cannot code a data set name and a file name (or DDname) or location parameters, or you will receive an error message.

FILE *dd-name* | DDNAME*dd-name*

Specifies the DD name. If you specify a file name (DDname), make sure that the appropriate JCL is coded for the entry.

DSNAME *dataset-name*

Specifies a data set name. If there is a period in the name,the name must be enclosed in quotes (single or double). For example, the data set TEST.LIB can be coded as 'TEST.LIB'.

The location parameters specify the element's location. Elements in CA Endevor SCM are identified by environment, system, subsystem, type, and stage (ID or number). Several actions require all or part of this information in the FROM clause. Whatever data you do not code in the syntax of the specific action must be entered in the SET FROM statement. You can use a name-mask with the system, subsystem, and type names, as well as with both stage indicators. Depending on the particular action, you may have a choice when entering a stage indicator (that is, ID or number). In this situation, the indicator is required, but you decide whether to enter an ID or stage number. If only one type of stage indicator appears in the SCL syntax, you must enter that specific value.

SITE *site-id*

Specifies the one-character ID for the location where CA Endevor SCM is installed.

ENVIRONMENT *env-name*

Specifies a one- to eight-character name for an environment in the software lifecycle.

SYSTEM *sys-name*

Specifies a one- to eight-character name for a system in the specified environment.

SUBSYSTEM *subsys-name*

Specifies a one- to eight-character name for a subgroup for the specified system.

TYPE *type-name*

Specifies a one- to eight-character name for the element type classification.

STAGE *stage-id*

Specifies a one-character stage ID that identifies a stage within the specified environment.

STAGE NUMBER *stage-no*

Specifies a one-character stage number that identifies the position of the stage within the specified environment.

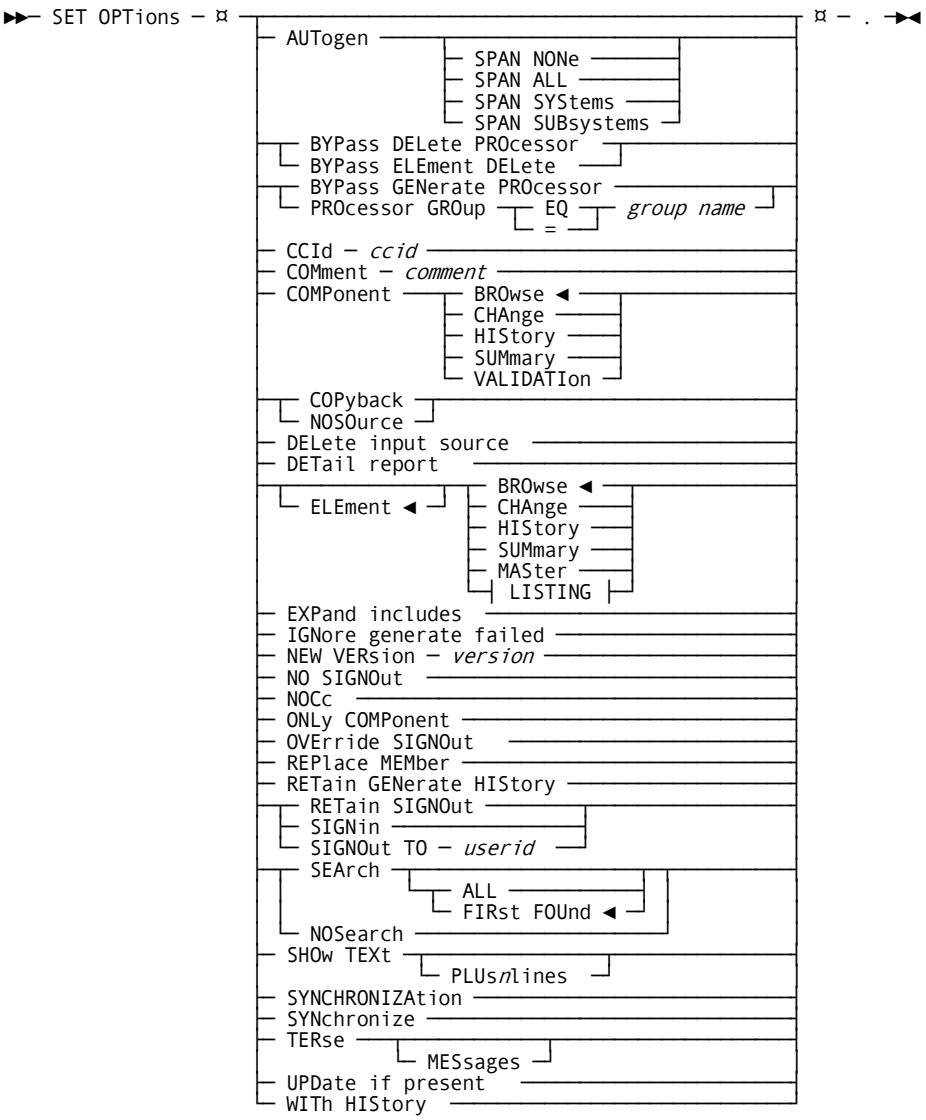
The Set Options Statement

The SET OPTIONS statement tells CA Endevor SCM to apply one or a series of options to all subsequent actions, until the next SET OPTIONS statement or a CLEAR OPTIONS statement is encountered, or processing ends. The exact options used depend on the action specified and the data you have entered in that element action statement:

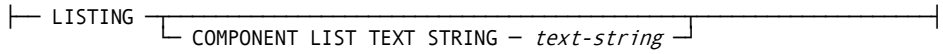
- Those options that do not apply to the action are ignored.
- If you enter a particular option in the element action statement and have coded that option in the SET OPTIONS statement, the entry in the action statement overrides the SET OPTIONS selection.

Set Options Syntax

The following is the syntax for set options action:



Expansion of LISTING



The SET OPTIONS statement tells CA Endevor SCM to apply one or a series of options to all subsequent actions, until the next SET OPTIONS statement or a CLEAR OPTIONS statement is encountered, or processing ends. The exact options used depend on the action you specify and the data you enter in that element action statement:

You can code the following options in the SET OPTIONS statement:

AUTOGEN | SPAN NONE| SPAN ALL | SPAN SYSTEMS | SPAN SUBSYSTEMS

Applicable for Add, Update, and Generate actions in batch requests. This option cannot be used in packages and the option does not work if Bypass Generate Processor is set. The Global Type Sequencing batch processing method must be enabled. Autogen only acts on components whose Types are listed in the Global Type Sequencing table. If the component's Type is not listed in the Global Type Sequencing table, the Autogen request is ignored. In addition, your site must have purchased and activated the CA Endevor Automated Configuration. Autogen can be specified alone or with various Span keyword options. "Autogen Span" is *not* a valid option. Valid options follow:

AUTOGEN

Generates all elements that use the component that is the target of the action. These *using* elements are generated at the target location that is specified in the SCL statement. If they do not exist at the target location, they are brought back to the target location as sourceless elements. An administrator can change the behavior of the Autogen feature, by activating AUTOGEN_SOURCE in the Optional Features Table (ENCOPTBL). When this option is activated, the Generate actions for the using elements are built with the Copyback, instead of the NoSource, option. For more information about sourceless elements, see the NoSource option description in Generate Syntax (see page 116).

Note: *Using* elements are elements that use the element that is the target of an Add , Update, or Generate action. For example, if Autogen is specified for copybook, COPYA, then the programs that use that copybook are known as using elements.

AUTOGEN SPAN NONE

Generates all elements that use the component being acted upon. This option has the exact same effect as the option "AUTOGEN."

AUTOGEN SPAN ALL

Generates using elements that are found in any System and Subsystem combinations within the Environment and Stage of the component's logical map.

AUTOGEN SPAN SYSTEMS

Generates using elements found in any System, provided the element's Subsystem name matches the name of the Subsystem of the target component. Only Systems found within the Environment and Stage of the component's logical map or higher up the map are searched. This option is different from the Autogen option in that it includes additional Systems with the same Subsystem name in the search.

AUTOGEN SPAN SUBSYSTEMS

Generates using elements from all Subsystems with the same-named System of the component specified. Only Subsystem found in the System of the target component within the Environment and Stage of the component's logical map or higher up the map are searched. This option is different from the Autogen option in that it includes additional Subsystems with the same System in the search.

The following restrictions apply to the SPAN options:

- Common libraries must be included in the generate processor concatenations of the Systems and Subsystems of the using elements. If the libraries are not included, then the Generate action does include the changes made to the component element when searching across Systems or Subsystems.
- The same Systems and Subsystems must exist in each Environment and Stage location. For example, if ELEMENTA is a using element in the PROD Environment, system A, then system A must exist in the DEV Environment also.
- SPAN only includes using elements from Systems or Subsystems located in the target Environment and higher up the map.

  SPAN does not include using elements from outside the Environment map.

BYPASS DELETE PROCESSOR

Tells CA Endevor SCM not to execute the delete processor for this element.

BYPASS ELEMENT DELETE

Tells CA Endevor SCM not to automatically delete the element in the FROM location after performing the action.

BYPASS GENERATE PROCESSOR

Indicates that CA Endevor SCM should not execute the generate processor for this element.

CCID ccid

Specifies an up to 12 character CCID.

COMMENT comment

Specifies an up to 40 character comment.

ELEMENT BROWSE | CHANGES | HISTORY | SUMMARY | MASTER | LISTING

Prints element information for or validates the element specified. The clause ELEMENT MASTER applies to both the Print action and the Validate action. The BROWSE, CHANGES, HISTORY, SUMMARY, and MASTER printouts provide the same information as their corresponding online panels.

The BROWSE, CHANGES, and SUMMARY options are not supported on sourceless elements. When encountered, the action is skipped and an information message is written to the action log. The MASTER option is supported for both sourced and sourceless elements.

The following rules apply to the ELEMENT clause:

- The ELEMENT keyword is implied, so it is not required to be specified with its parameters. For example, if CHANGES is specified, then ELEMENT CHANGES is implied.
- The parameters BROWSE, CHANGES, HISTORY, and SUMMARY can be used with either the COMPONENT clause or the ELEMENT clause.
- Generally, only one COMPONENTS clause or one ELEMENT clause is used on a statement. If a COMPONENTS clause and an ELEMENT clause, or multiple COMPONENTS and ELEMENT clauses are specified in the same statement, then only the last such clause in the statement is used.

  For the PRINT statement, using a SET OPTIONS MASTER COMP statement will execute both the element master and component browse action options. However, a PRINT statement using a SET OPTIONS ELEMENT HISTORY COMP statement will only execute component history option, which was the last option specified.
- Only one parameter can be specified in a COMPONENTS clause or an ELEMENT clause.
- BROWSE is the default for ELEMENT or COMPONENT.

  For example:

  - If neither COMPONENT or ELEMENT is specified with one of the BROWSE, CHANGES, HISTORY, and SUMMARY parameters, then ELEMENT is the default.
  - If neither COMPONENT or ELEMENT is specified and none of the parameters are specifed, then ELEMENT BROWSE is the default.
  - If ELEMENT is specified without any parameters, then ELEMENT BROWSE is the default.

- If COMPONENT is specified without any parameters, then COMPONENT BROWSE is the default.
- If any of the parameters are specified without ELEMENT or COMPONENT, then element information is returned, because the ELEMENT keyword is the default. For example, if CHANGES is specified, then ELEMENT CHANGES is the implied.

Only one of the following parameters BROWSE, CHANGES, HISTORY, SUMMARY, MASTER, or LISTING can be specified with ELEMENT.

BROWSE

Prints information about the current element, indicating the level at which each line was added. If you specify a particular level, CA Endevor SCM prints the source for that level. This is the default.

CHANGES

Prints all the changes-inserts and deletes-made to the element at the level specified. If you do not specify a level in the LEVEL clause, changes for the current level of the element are shown.

HISTORY

Prints all lines that have ever been in the element, noting the level at which the line was added, changed, or deleted. If you specify a level in the LEVEL clause, CA Endevor SCM prints history for that level.

SUMMARY

Prints a summary line of data for each level of the element specified, and includes information appropriate to that level (for example, the number of inserts and the number of deletes).

MASTER

Prints or Validates Master Control File information for the element, as well as current data pertaining to the element (such as last processor, processor return codes, current version or level, and so on).

LISTING

Prints the element's associated output listing. The following parameter is optional:

COMPONENT LIST TEXT STRING *text-string*

Searches all component output data sets for the specified string.

COMPONENT BROWSE | CHANGES | HISTORY | SUMMARY | VALIDATION

Prints component information or validates components for the element specified. The COMPONENT keyword is required to be specified for this clause. The BROWSE, CHANGES, HISTORY, SUMMARY, printouts provide the same information as their corresponding online panels. The VALIDATION parameter applies to the Validate action.

Note: For the Validate action, the component validation option can be specified differently depending upon whether it is specified on the Validate action statement or in the Set Options statement. The Set Options statement requires both keywords COMPONENT and VALIDATION. However, the component validation option can also be set by specifiying OPTION COMP in a Validate action statement. Also, although specifying SET OPTION COMP does not set the component validation option for Valdiate actions, it does set the component browse option for Print actions.

The CA Endevor SCM Automated Configuration option is required to use the COMPONENT keyword. CA Endevor SCM prints as much information as is available for the component list. For example, if you code COMPONENTS CHANGES but there were no changes to the output components section, that section would not appear in the associated listing.

The BROWSE, CHANGES and SUMMARY options are not supported on sourceless elements. When encountered, the action is skipped and an information message is written to the action log.

Only one of the following options can be specified with COMPONENT:

BROWSE

Prints the current component list source, indicating the level at which each line was added. If you specify a particular level, CA Endevor SCM prints the source for that level. This is the default.

CHANGES

Prints all the changes-inserts and deletes-made to the component list at the level specified. If you do not specify a level in the LEVEL clause, changes for the current level of the element are shown.

HISTORY

Prints all lines that have ever been in the component list source, noting the level at which the line was added, changed, or deleted. If you specify a level in the LEVEL clause, CA Endevor SCM prints history for that level.

SUMMARY

Prints a summary line of data for each level of the element or component list specified, and includes information appropriate to that level (for example, the number of inserts and the number of deletes).

VALIDATION

Checks to make sure that all the components exist and are valid. This option applies to the Validate action.

COPYBACK

Tells CA Endevor SCM to copy the current level of an element back to the target stage for a GENERATE action, prior to generating the element. COPYBACK cannot be specified with NOSOURCE.

DELETE INPUT SOURCE

Tells CA Endevor SCM to delete a member from the library in which it originated.

DETAIL REPORT

Tells CA Endevor SCM to provide detail information in the Execution Report. CA Endevor SCM, by default, lists only those elements matching the selection criteria you specify. If you select the DETAIL REPORT option, every element searched is listed in the report-whether or not a match is found.

EXPAND INCLUDES

Tells CA Endevor SCM to expand INCLUDE statements when the element is copied to a source output library.

IGNORE GENERATE FAILED

Enables processing to continue when the generate and/or move processors associated with a particular element have failed.

JUMP

Tells CA Endevor SCM to notify the user if an element exists at an intermediate, non-map stage between the source and target stages of a MOVE.

NEW VERSION

Lets you to assign a different version number to the TO location element. Simply enter the number (1-99 inclusive, leading zeros optional) that you want to use.

NO SIGNOUT

Tells CA Endevor SCM to retrieve an element without signing it out.

NOCC

Suppresses the default printing of a header on each page of output. For the PRINT ELEMENT MASTER output, other line feed characters in column 1 of the output are replaced with a blank space.

NOSOURCE

When the target location has a sourced element, the element is generated in place.

When the target location has a sourceless element, the element is generated at the target location using the source of the first occurrence of the element found up the map.

When the element does not exist at the target location, the element is generated at the target location using the source of the first occurrence of the element found up the map. The source is not fetched to the target. The MCF element created at the target location will contain data similiar to a fetched back element except that the element base and delta name fields will be blank and the record will be marked as a sourceless element.

NOSOURCE cannot be used with COPYBACK. It is not necessary to specify the SEARCH option with NOSOURCE, because NOSOURCE implies SEARCH. If SEARCH is specified with NOSOURCE, the SEARCH parameter is ignored.

ONLY COMPONENTS

Lets you to delete the component lists for an element, but not the element itself.

OVERRIDE SIGNOUT

Enables you to access an element that has been signed out to a user ID other than your own. Use OVERRIDE SIGNOUT with caution to avoid regressing changes made by another user.

PROCESSOR GROUP EQUAL/EQ/= group name

Specifies a 1- to 8-character processor group name.

REPLACE MEMBER

Tells CA Endevor SCM to replace an existing member in a target library with the element specified in the element action statement.

RETAIN GENERATE HISTORY

Retains the Master Control File last generate information. This option when used with the Transfer to Archive action can be useful when converting elements to a new delta type.

RETAIN SIGNOUT

Tells CA Endevor SCM to retain the current signout for an element.

SEARCH [ALL | FIRST FOUND] | NOSEARCH

Specifies a logical or physical search of the software inventory that is based on the C1DEFLTS table Environment definitions. After an Environment Stage is matched, the Stage's Master Control File is searched based on other criteria specified in the Endevor action.

SEARCH [ALL | FIRST FOUND]

Specifies a logical search of the software inventory that is based on the mapping of Environment Stages as defined in the C1DEFLTS table. The search begins with the Environment Stage that is specified on the element action From clause and continues along the map. On the From clause, the Environment name must be explicit. Name-masking is supported on the Stage ID.

The ALL and First Found options are ignored by all actions, except the Alter action.

ALL— Identifies all elements that are found in the logical search.

FIRST FOUND— Identifies the first element that is found in the logical search.

NOSEARCH

Specifies a physical search of the software inventory that is based on an exact match to the C1DEFLTS table Environment definitions, regardless of mapping. The search is limited to the Environment Stage that is specified on the element action From clause. Name-masking is supported on the Environment name (for the Alter action only) and on the Stage ID. If the Environment or Stage is name-masked, more than one Environment Stage can meet the selection criteria.

SHOW TEXT PLUS n LINES

Tells CA Endevor SCM to print the line of source code that contains a specified text string, plus a designated number of lines of code before and after the text string.

SIGNIN

Lets you to override a RETAIN SIGNOUT or SIGNOUT TO clause in a SET OPTIONS statement.

SIGNOUT TO

Lets you to sign out or reassign an element to another user.

SYNCHRONIZATION

Checks to make sure that an out-of synch condition is not caused by the action being performed. This option applies to the Validate action.

SYNCHRONIZE

Tells CA Endevor SCM to create a sync level at a target location when the base level of an element at a source location is not the same as the current level of that element at the target location.

TERSE MESSAGES

Limits the amount of message detail that is written to the C1MSGS1 report file for those messages that result from the Validate action.

UPDATE IF PRESENT

Automatically changes an ADD action to an UPDATE action if an element currently exists in the entry stage. This option essentially allows you to add the element to the entry stage.

WITH HISTORY

Tells CA Endevor SCM to preserve change history for an element when transferring or moving that element.

Actions and the Set Options Statement

The following table indicates the actions for which you can code each option, and provides notes on the use of each option.

Options	Actions	Notes
AUTOGEN, AUTOGEN SPAN NONE, AUTOGEN SPAN ALL, AUTOGEN SPAN SYSTEMS, AUTOGEN SPAN SUBSYSTEMS	ADD, UPDATE, GENERATE	The product option CA Endevor Automated Configuration is required to use the AUTOGEN options. Global Type Sequencing must be enabled. Cannot be used with BYPASS GENERATE PROCESSOR.
BYPASS DELETE PROCESSOR	TRANSFER	Cannot be used with BYPASS ELEMENT DELETE. Cannot be used for transfer from external data set to CA Endevor SCM.
BYPASS ELEMENT DELETE	ARCHIVE, MOVE, TRANSFER	Cannot be used with BYPASS DELETE PROCESSOR. Cannot be used for transfer from external data set to CA Endevor SCM.
BYPASS GENERATE PROCESSOR	ADD, RESTORE, TRANSFER, UPDATE	Cannot be used for transfer from CA Endevor SCM to external data set. Cannot be used with PROCESSOR GROUP EQUAL.

CCID *ccid*	ADD, ALTER, ARCHIVE, DELETE, GENERATE, MOVE, RESTORE, RETRIEVE, TRANSFER, UPDATE, VALIDATE	
COMMENT *comment*	Same as for CCID	
[COMPONENT] BROWSE	PRINT	The option CA Endevor SCM Automated Configuration is required to use [COMPONENT]. One clause (BROWSE, CHANGE, HISTORY, SUMMARY) allowed per statement.
[COMPONENT] CHANGES	PRINT	Same as [COMPONENT] BROWSE.
[COMPONENT] HISTORY	PRINT	Same as [COMPONENT] BROWSE
[COMPONENT] SUMMARY	PRINT	Same as [COMPONENT] BROWSE
COMPONENT VALIDATION	VALIDATE	Same as [COMPONENT] BROWSE
COMPONENT LIST TEXT STRING	PRINT	The option CA Endevor SCM Automated Configuration is required.
COPYBACK	GENERATE	Cannot be used with NOSOURCE.
DELETE INPUT SOURCE	ADD, UPDATE	
DETAIL REPORT	LIST	
[ELEMENT] MASTER	PRINT, VALIDATE	The keyword ELEMENT is required for the VALIDATE action.
[ELEMENT] LISTING	PRINT	The option CA Endevor SCM Automated Configuration is required.
[ELEMENT] BROWSE	PRINT	
[ELEMENT] CHANGES	PRINT	
[ELEMENT] HISTORY	PRINT	
[ELEMENT] SUMMARY	PRINT	
EXPAND INCLUDES	RETRIEVE	

IGNORE GENERATE FAILED	TRANSFER	Cannot be used for transfer from external data set to CA Endevor SCM.
JUMP	MOVE	
NEW VERSION version number	ADD, RESTORE, TRANSFER	Cannot be used for transfer from CA Endevor SCM to external data set.
NO SIGNOUT	RETRIEVE	
NOCC	PRINT	
NOSEARCH	ALTER, GENERATE, LIST, PRINT, RETRIEVE	Cannot be used with SEARCH.
NOSOURCE	GENERATE	Cannot be used with COPYBACK.
ONLY COMPONENTS	DELETE	
OVERRIDE SIGNOUT	ADD, ALTER, ARCHIVE, DELETE, GENERATE, RETRIEVE, SIGNIN, TRANSFER, UPDATE	
PROCESSOR GROUP EQUAL GROUP NAME	ADD, GENERATE, RESTORE, TRANSFER, UPDATE	Cannot be used with BYPASS GENERATE PROCESSOR.
REPLACE MEMBER	COPY, LIST, RETRIEVE	
RETAIN GENERATE HISTORY	RESTORE	
RETAIN SIGNOUT	MOVE, TRANSFER	Cannot be used with SIGNIN or SIGNOUT TO.
SEARCH	ALTER, GENERATE, LIST, PRINT, RETRIEVE	Cannot be used with NOSEARCH.
SHOW TEXT [PLUS n LINES]	LIST	
SIGNIN	MOVE, TRANSFER	Cannot be used with RETAIN SIGNOUT or SIGNOUT TO

SIGNOUT TO USERID	MOVE, SIGNIN, TRANSFER	Cannot be used with SIGNIN or RETAIN SIGNOUT
SYNCHRONIZATION	VALIDATE	
SYNCHRONIZE	MOVE, TRANSFER	
TERSE MESSAGES	VALIDATE	
UPDATE IF PRESENT	ADD	
WITH HISTORY	MOVE, TRANSFER	

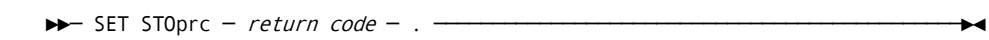
The Set STOPRC Statement

The SET STOPRC statement provides a control for processing during batch execution. Prior to executing the job stream, CA Endevor SCM checks for the SET STOPRC statement. If more than one statement has been coded, the return code entered in the last statement found is used.

During execution, CA Endevor SCM checks the CA Endevor SCM return code (NDVR RC) for the current action before proceeding with the next action.

Set STOPRC Syntax

The following is the syntax for set to action:



The STOPRC statement identifies your highest acceptable return code for the current action processing. You can specify a return code value in the range of 4 through 16. If you do not enter a STOPRC value, CA Endevor SCM operates as if a STOPRC of 16 has been coded.

If the CA Endevor SCM return code is equal to or exceeds the return code entered in the STOPRC statement, CA Endevor SCM stops processing, the remaining actions are not executed, and CA Endevor SCM returns with return code of 12 and error message C1G0236E. If CA Endevor SCM encounters a fatal error while executing an action, it does not check for STOPRC and returns with return code of 16 and error message C1G0210S.

If during concurrent action processing, an action request that was routed to an action processing region returns a return code that exceeds the STOPRC value, then no more requests are dispatched. Concurrent action processing ceases and any remaining 'in-flight' requests are allowed to terminate. Because several requests are in-flight at the same time, some requests may continue to completion even though the STOPRC value was reached.

The Set To Statement

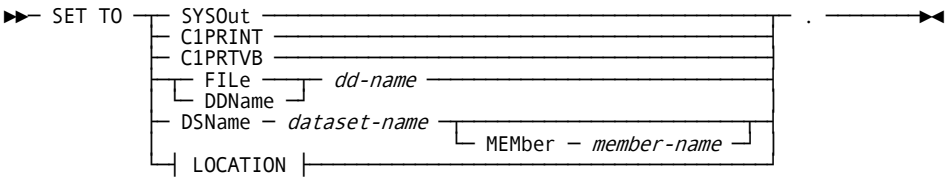
The SET TO statement applies to each element action that uses-but does not contain all or part of-a TO clause, and remains in effect until CA Endevor SCM encounters another SET TO statement or a CLEAR TO statement, or when processing ends.

The exact information used from the SET TO statement depends on both the specific action and the data you have entered in that element action statement. What you enter in the action's TO clause overrides that particular entry in the SET TO statement. For example, you code all CA Endevor SCM location information (environment, system, subsystem, type, and stage ID or stage number) in the SET TO statement. Then, when coding an UPDATE statement, you enter a different subsystem. CA Endevor SCM determines the TO location by applying all SET TO information except for subsystem, which is taken from the UPDATE statement.

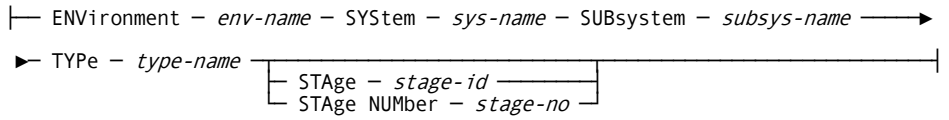
Note: The SET TO information you enter differs from action to action. For more information, see the chapter "Processing Element Actions (see page 85)." Remember that you cannot use a name-mask with any TO location field names.

Set To Syntax

The following is the syntax for set to action:



Expansion of LOCATION



SET TO SYSOUT

SYSOUT applies to the LIST action only. Normally when you execute the LIST action, CA Endevor SCM lists the action cards in both the listing (Execution Report) and the location you have indicated in the TO clause. If you do not enter any information in the TO clause for the LIST action, CA Endevor SCM checks the SET TO statement for information. If the appropriate information has not been entered in the SET TO statement or the SET TO statement indicates only SYSOUT, CA Endevor SCM defaults to SYSOUT alone.

When SYSOUT alone is selected, the action cards requested in the LIST action are printed immediately after the LIST request, as part of the listing. You cannot perform any editing on these action cards because they are available only in the printout. If you have indicated another location (such as a library) in the TO clause, however, you can access, and therefore edit, the action cards generated.

SET TO C1PRINT

C1PRINT applies to the PRINT action only. If you do not enter any information in the TO clause for the PRINT action, CA Endevor SCM checks the SET TO statement for information. If the appropriate information has not been entered in the SET TO statement or the SET TO statement indicates C1PRINT, CA Endevor SCM defaults to C1PRINT and prints the specified element or member in a listing. When using C1PRINT, be sure you have included the appropriate JCL. See the following examples:

- To send your output to the queue, code the following:

  ```
  //C1PRINT DD SYSOUT=*
  ```

- To send your output to an existing file, code the following:

  ```
  //C1PRINT DD DISP=SHR,DSN=filename
  ```

SET TO C1PRTVB

Prints elements that have records that are longer than 121 characters to a location external to CA Endevor SCM (for example, a library, sequential file, or PDS). You must have previously allocated the C1PRTVB data set appropriately. The recommended DCB for C1PRTVB is LRECL=27994,BLKSIZE=0,RECFM=VB. However, if the record size of any record is longer than 27978, code a larger record length. You must code a sufficiently long LRECL for the output file. The LRECL size should be at least 16 bytes longer than the longest record of the element. For example, allocate the data set as follows:

```
//C1PRTVB  DD DISP=(,CATLG),DSN=my.C1PRTVB,
//         SPACE=TRK,(5,5),UNIT=SYSDA,
//         DCB=(RECFM=VB,LRECL=27994,BLKSIZE=0)
```

SET TO FILE (DDNAME) *dd-name*

DSNAME *dataset-name*

When the TO location for the element is external to CA Endevor SCM (for example, a library, sequential file, or PDS), you can enter either a file name (or DDname) or a data set name in the TO clause.

- When you enter a file name (DDname), be sure that the appropriate JCL is coded for the entry.
- When you enter a data set name, be sure to enclose the name in quotes (single or double) if there is a period in the name; for example, the data set TEST.LIB must be coded as **'TEST.LIB'**.

Note: You cannot code both a file name (or DDname) and a data set name.If you do, you receive an error message. You also receive an error message if you enter CA Endevor SCM location information along with a data set name.

SET TO ENVIRONMENT ***env-name***

SYSTEM ***sys-name***

SUBSYSTEM ***subsys-name***

TYPE ***type-name***

STAGE ***stage-id***

STAGE NUMBER ***stage-no***

Elements in CA Endevor SCM are identified by environment, system, subsystem, type, and stage (ID or number). Several actions require all or part of this information in the TO clause. Whatever data you do not code in the syntax of the specific action must be entered in the SET TO statement.

A brief definition of each identifier follows.

- ENVIRONMENT-The functional areas within an organization. Environment names can be up to **8** characters in length.
- SYSTEM-The applications at a site. System names can be up to **8** characters in length.
- SUBSYSTEM-Specific applications within a system. Subsystem name can be up to **8** characters in length.
- TYPE-Categories of source code. Type names can be up to **8** characters in length.
- STAGE-A stage in a software life cycle. You refer to stages i n SCL statements by one of the following:
  - STAGE ID-A **1**-character, alphanumeric stage identifier.
  - STAGE NUMBER-Either **1** or **2**. Indicates the position of a stage within the current environment.

Note: For more information about each term, see the *User Guide*.

SET TO MEMBER NAME

SET TO MEMBER applies only to the LIST action. If you do not enter a member name in the LIST action, CA Endevor SCM checks the related SET TO statement for a member name. If a member name has not been coded, the system defaults to SYSOUT and the list is produced in the listing immediately following the request.

Note: If this statement is used for any other action other than LIST it will be ignored.

The Set Where Statement

The SET WHERE statement applies to each element action that uses-but does not contain all or part of-a WHERE clause, and remains in effect until the system encounters another SET WHERE statement or a CLEAR statement, or processing ends. The exact information used from the SET WHERE statement depends on both the specific action and the data you have entered in that element action statement. What you enter in the action's WHERE clause overrides that particular entry in the SET Where statement.

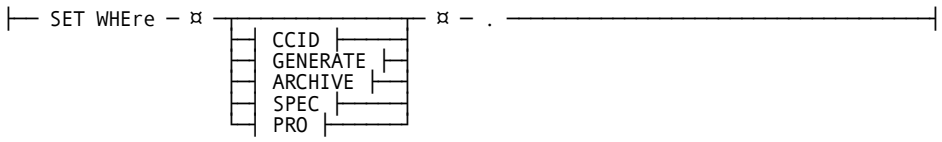
SET WHERE differs from the SET BUILD, SET FROM, and SET TO statements in that the WHERE (and consequently the SET WHERE) clause is optional. If you do not enter WHERE information for a specific action and a SET WHERE statement has not been coded, the system continues processing; you do not receive an error message nor does processing terminate.

The WHERE clause is most useful when you are using a name-mask, as it further qualifies the criteria you have entered for the element(s). When you use a name-mask, the designated action is performed for only those elements matching the WHERE criteria entered (along with any other qualifying data entered).

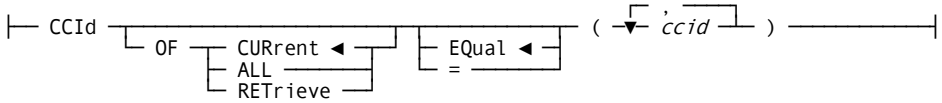
Each clause is explained in the following section.

Note: To determine which WHERE information you can enter for each request, see the chapter "Processing Element Actions."

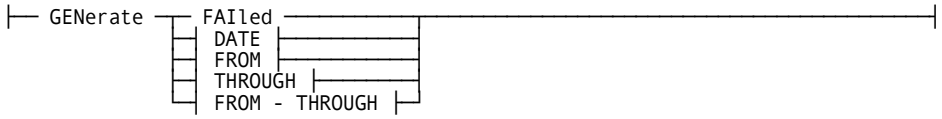
Set Where Syntax



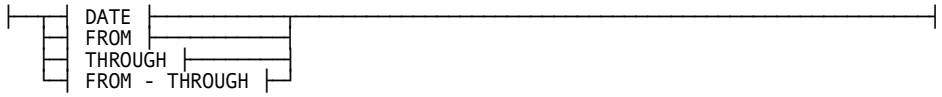
Expansion of CCID



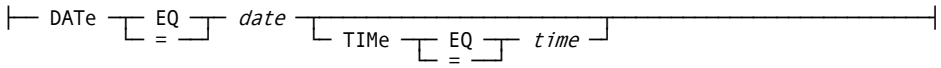
Expansion of GENERATE



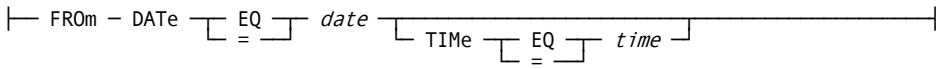
Expansion of ARCHIVE



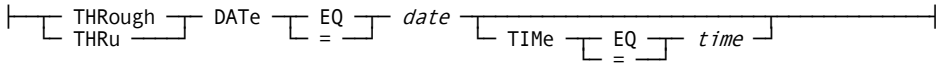
Expansion of DATE



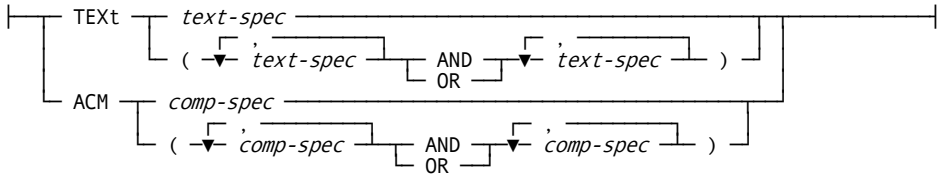
Expansion of FROM



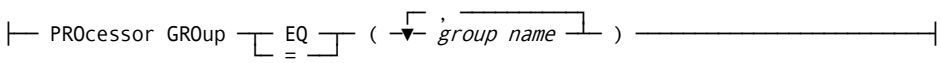
Expansion of THROUGH



Expansion of SPEC



Expansion of PRO



SET WHERE CCID

The SET WHERE statement applies to each element action that uses-but does not contain all or part of-a WHERE clause, and remains in effect until the system encounters another SET WHERE statement or a CLEAR statement, or processing ends. There are two forms of WHERE CCID SCL:

WHERE CCID ccid-Limits processing to only those elements that match one of the CCIDs coded.

WHERE CCID OF ccid-Also limits the processing to those elements that match one of the supplied CCIDs. With this SCL, however, you can indicate where you want CA Endevor SCM to look for the CCID(s):

- **CURRENT-**Tells CA Endevor SCM to look through the CCID fields in the MCF (Master Control File) to find a specified CCID(s). This is the default.
- **ALL-**Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s).
- **RETRIEVE-**Tells CA Endevor SCM to use the CCID in the Master Control File's RETRIEVE CCID field.

You can use a name-mask with the CCID.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The examples below illustrate the two forms of WHERE CCID SCL.

```
Example 1: WHERE CCID EQ PROJ00*
Example 2: WHERE CCID OF CURRENT (PROJ001, PROJ002, PROJ004)
Example 3: WHERE CCID OF ALL (PROJ00*)
```

SET WHERE GENERATE

WHERE GENERATE SCL allows you to set a generation date and, optionally, time as a selection criterion. There are five possible forms for this clause:

- **WHERE GENERATE FAILED-**Tells CA Endevor SCM to list only those elements for which the generate processor failed.
- **WHERE GENERATE DATE mm/dd/yy TIME hh:mm-**Tells CA Endevor SCM to select only those elements with this generate date, and optionally, this time stamp.
- **WHERE GENERATE FROM DATE mm/dd/yy TIME hh:mm-**Tells CA Endevor SCM to select all elements with a generate date and, optionally, a time stamp on or after the specified date and time stamps.
- **WHERE GENERATE THROUGH DATE mm/dd/yy TIME hh:mm-**Tells CA Endevor SCM to select all elements with a generate date and, optionally, a time stamp earlier than and including the specified date and time stamp.
- **WHERE GENERATE FROM DATE mm/dd/yy TIME hh:mm THROUGH DATE mm/dd/yy TIME hh:mm**-Tells CA Endevor SCM to select only those elements with a generate date, and optionally, time stamps within the specified range.

The date(s) must be in *mm/dd/yy* format (leading zeros are not required). The time(s) must be in *hh:mm* format. If you enter a time in this clause, you must enter a date.

SET WHERE ARCHIVE

WHERE ARCHIVE SCL allows you to set an archive date and, optionally, time as a selection criteria. There are four possible forms for this clause:

- **WHERE ARCHIVE DATE mm/dd/yy TIME hh:mm-**Tells CA Endevor SCM to select only those elements with this archive date, and optionally, this time stamp.
- **WHERE ARCHIVE FROM DATE mm/dd/yy TIME hh:mm-**Tells CA Endevor SCM to select all elements with an archive date and, optionally, a time stamp on or after the specified date and time stamps.

- **WHERE ARCHIVE THROUGH DATE mm/dd/yy TIME hh:mm-**Tells CA Endevor SCM to select all elements with an archive date and, optionally, a time stamp earlier than and including the specified date and time stamp.
- **WHERE ARCHIVE FROM DATE mm/dd/yy TIME hh:mm THROUGH DATE mm/dd/yy TIME hh:mm-**Tells CA Endevor SCM to select only those elements with an archive date and, optionally, a time stamp within the specified range.

The date(s) must be in *mm/dd/yy* format (leading zeros are not required). The time(s) must be in *hh:mm* format. If you enter a time in this clause, you must enter a date.

SET WHERE TEXT

WHERE TEXT SCL limits a list to elements that contain (or do not contain) one or more specified 1- to 70-character text strings.

For example:

- This example tells CA Endevor SCM to list all elements containing the text string "WO9- LINKAGE":

  ```
  WHERE TEXT 'WO9-LINKAGE'
  ```

- This example tells CA Endevor SCM to list all elements that contain the text strings "COPY COPY005" and "COPY COPY010" between columns 8 and 40 of the element source:

  ```
  WHERE TEXT ('COPY COPY005' COLUMN 8 40 AND 'COPY COPY010' COLUMN 8 40)
  ```

- This example tells CA Endevor SCM to list all elements that do not contain the text string "REMARKS" between columns 8 and 15 of the element source:

  ```
  WHERE TEXT DOES NOT CONTAIN 'REMARKS' COLU.MN 8 15
  ```

- This example tells CA Endevor SCM to list all elements that contain either the text string "M605SUB" or the text string "M607SUB" and do not contain the text string "M606SUB":

  ```
  WHERE TEXT (('M605SUB' OR 'M607SUB')AND DOES NOT CONTAIN 'M606SUB')
  ```

Note: The WHERE TEXT EQUAL clause cannot be used withthe WHERE ACM clauses.

SET WHERE ACM

WHERE ACM SCL limits a list to component lists containing (or not containing) the designated component name. Wildcards are acceptable in the component name specification.

There are four clauses:

- **WHERE INPUT COMPONENT** tells CA Endevor SCM to list only input components matching your entry. This is the default.
- **WHERE RELATED INPUT COMPONENT-**tells CA Endevor SCM to list only related input components matching your entry.
- **WHERE OUTPUT COMPONENT** tells CA Endevor SCM to list only output components matching your criteria.
- **WHERE RELATED OUTPUT COMPONENT-**tells CA Endevor SCM to list only related output components matching your entry.
- **WHERE PROCESSOR COMPONENT** tells CA Endevor SCM to list only processor components matching the criteria.
- **WHERE ALL COMPONENT** tells CA Endevor SCM to list for matches within all three types of components.

Additional selection criteria for the component includes the following clauses.

- **THROUGH (THRU)** *comp-name*-tells CA Endevor SCM to list elements within a specific range of component names. The range begins with the component name coded in the WHERE COMPONENTS clause, and encompasses all components up to and including the component specified in this clause. Wildcards are acceptable in the component name specification.
- **VERSION** *version*-tells CA Endevor SCM to list elements containing components with a specific 1- to 99-character version number. The version number of the component may differ from the version number of the element with which it is associated.
- **LEVEL** *level* tells-CA Endevor SCM to list elements containing components with a specific 0- to 99-character level number. The level number of the component can differ from the level number of the element with which it is associated.
- **ENVIRONMENT** *env name*-tells CA Endevor SCM to list elements with components located in the specified environment. If you provide an environment name, you must also provide the following information:
  - **SYSTEM-**1 to 8 characters
  - **SUBSYSTEM-**1 to 8 characters
  - **TYPE-**1 to 8 characters
  - **STAGE NUMBER-**either **1** or **2**

- **FILE (DDNAME) ddname-**tells CA Endevor SCM to list elements whose:
  - Input components originated from the specified DDname
  - Output components were written to the specified DDname
  - Components were produced by a processor step specified by and associated with the designated DDname
- **DSNAME data set name**-tells CA Endevor SCM to list elements whose:
  - Input components originated from the specified data set
  - Output components were written to the specified data set
  - Components were produced by a processor step specified by and associated with the designated data set

WHERE ACM comp spec {AND/OR} comp spec

This clause allows you to provide compound component selection criteria, using the same options as described above.

Note: The WHERE ACM clauses cannot be used with the WHERE TEXT clause.

WHERE PROCESSOR GROUP

WHERE PROCESSOR GROUP SCL allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The examples below illustrate the use of this clause.

```
Example 1: WHERE PROCESSOR GROUP (COBVS, COBII)
Example 2: WHERE PROCESSOR GROUP (COBV)
```

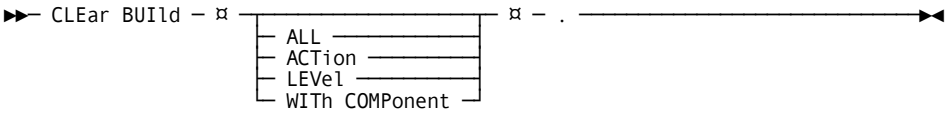
Clear Statements

A CLEAR statement clears the information that has been designated by a SET statement. The CLEAR statement must be in the same syntax as the SET statement to which it applies, and must be entered (at some point in your code) after that SET statement. The CLEAR statement affects all syntax following it until a new SET statement is encountered or processing ends. The CLEAR statement does not affect the related information coded within each individual element action's syntax. Because these statements are not executed, no source or output management is involved.

The Clear Build Statement

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

Clear Build Syntax



You can code the following options in the CLEAR BUILD statement.

ALL

Clears every selection designated in the related SET BUILD clause-action, level, and WITH COMPONENTS (if applicable).

ACTION

Clears the related SET BUILD ACTION clause, no matter which action is coded in that clause.

LEVEL

Clears the level selection designated in the related SET BUILD LEVEL clause.

WITH COMPONENTS

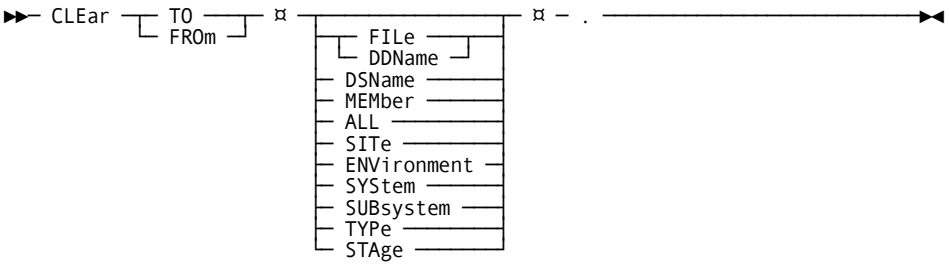
Clears the related SET BUILD WITH COMPONENTS clause.

Note: WITH COMPONENTS pertains to the CA Endevor SCM ACM product only.

The Clear To and From Statements

The CLEAR TO and CLEAR FROM statements clear information from previously coded SET TO and SET FROM statements.

Clear To and From Syntax



You can enter the following values in the CLEAR TO and CLEAR FROM statements:

FILE/DDNAME

Clears the related SET TO/FROM FILE (DDNAME) clause.

DSNAME

Clears the related SET TO/FROM DSNAME clause.

MEMBER

Clears the related SET TO/FROM MEMBER clause.

ALL

Clears all clauses entered for the related SET statement(s).

SITE

Clears the related SET FROM SITE (site ID) clause.

ENVIRONMENT

Clears the related SET TO/FROM ENVIRONMENT clause.

SYSTEM

Clears the related SET TO/FROM SYSTEM clause.

SUBSYSTEM

Clears the related SET TO/FROM SUBSYSTEM clause.

TYPE

Clears the related SET TO/FROM TYPE clause.

STAGE

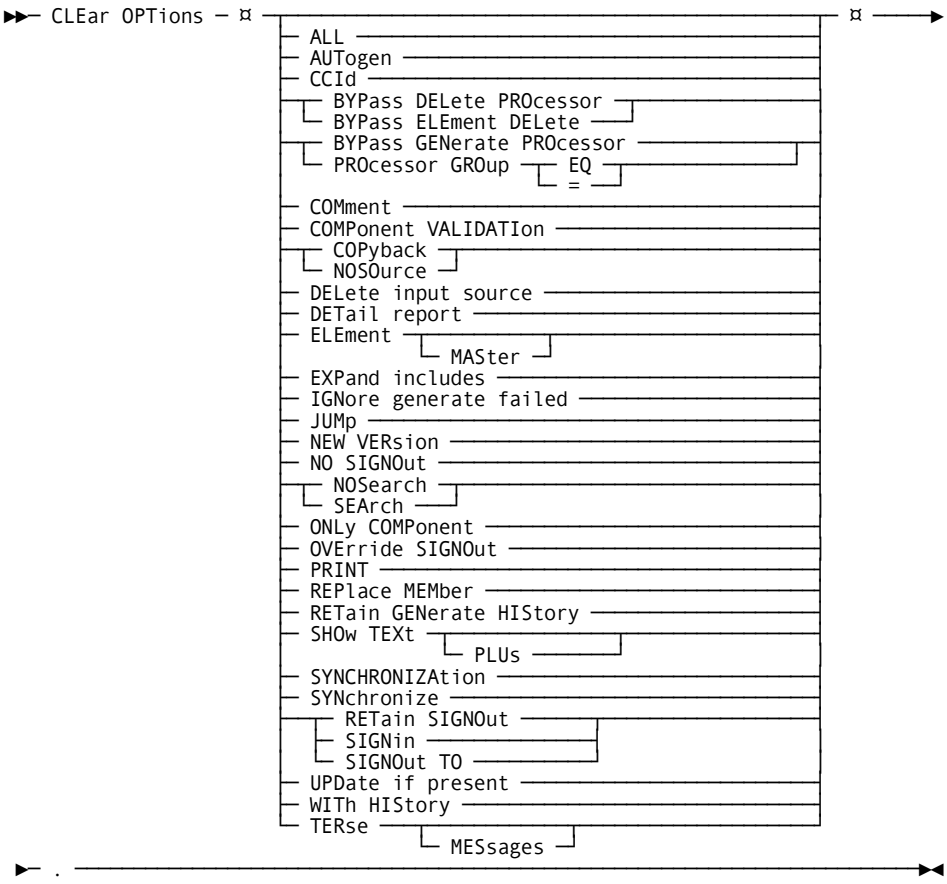
Clears the related SET TO/FROM STAGE (ID) or SET TO/FROM STAGE NUMBER clause.

The Clear Options Statement

The CLEAR OPTIONS statement clears any "matching" SET OPTIONS statement coded previously.

Clear Options Syntax

The syntax for the clear options action is shown in the following diagram:



CLEAR OPTIONS

Clears the options specified. These options were previously set in the Set Options statement. For information about these options, see Set Options Syntax (see page 57). Additional information about the keywords you can specify on the Clear Option statement are provided next:

ALL

Clears all the options previously set.

AUTOGEN

Clears Autogen including any Span options that were previously set.

COMPONENT VALIDATION

Clears the component validation option for the Validate action only. This option does not apply for the Print action. The statement CLEAR OPTIONS COMP parses without errors; however, it does not clear any of the component options.

ELEMENT MASTER

Clears the element master option for the Validate action only. This option does not apply for the PRINT action.

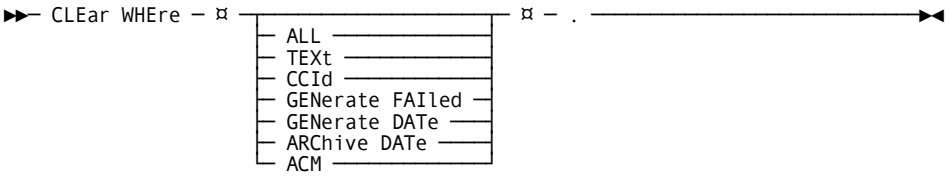
PRINT

Clears all the print options: NOCC, COMPONENT, the display type (browse, change, history, summary, master, and listing) and the COMPONENT LIST TEXT STRING value. PRINT is the only keyword that will clear any of the options for the Print action; CLEAR ELEMENT MASTER or CLEAR COMPONENT have no effect for Print actions.

The Clear Where Statement

The CLEAR WHERE statement clears all related SET WHERE clauses coded previously.

Clear Where Syntax



You can enter the following values in the CLEAR WHERE statement:

ALL

Clears all SET WHERE statements previously coded.

TEXT

Clears the related SET WHERE TEXT EQUALS clause.

CCID

Clears the related SET WHERE CCID clause.

GENERATE FAILED

Clears the related SET WHERE GENERATE FAILED clause.

GENERATE DATE

Clears the related SET WHERE GENERATE DATE (and GENERATE TIME) clause.

ARCHIVE DATE-

Clears the related SET WHERE ARCHIVE DATE (and ARCHIVE TIME) clause.

ACM

Clears all information coded in relation to the SET WHERE COMPONENTS EQUAL clause, including:

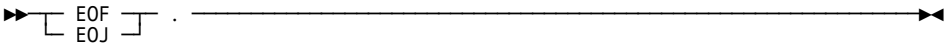
- Type of component (input, output, processor, all).
- THROUGH, VERSION, LEVEL in a WHERE COMPONENTS EQUAL clause.
- Component inventory location (environment, system, subsystem, type, and stage number).
- File (DDname) or data set name.

The EOF (EOJ) Statement

The EOF (EOJ) statement tells CA Endevor SCM to stop parsing the SCL syntax at a particular point. For example, if you have listed two actions and want to perform only the first action, you would enter EOF. (or EOJ.) immediately after the last line of the first action (or immediately before the first line of the second action).

If you do not use the EOF (EOJ) statement, you need to manually delete the actions (lines of code) you do not want performed.

EOF (EOJ) Syntax



EOF (EOJ)

Simply code either **EOF** or **EOJ** in the appropriate place in the syntax.

Chapter 4: Processing Element Actions

This section contains the following topics:

SCL Coding Conventions

Most of the actions for which you can code SCL can also be accessed in foreground or batch mode. A discussion of the processing flow for these actions in foreground can be found in the *User's Guide*.

A strict coding order applies to the THROUGH, VERSION, and LEVEL clauses, as follows:

- When coding the THROUGH clause, it must immediately follow the initial action clause. If you enter a THROUGH clause, however, you cannot specify a level for the action.
- When coding the VERSION clause, it must immediately follow the THROUGH clause. If a THROUGH clause has not been entered, the VERSION clause must immediately follow the initial action clause.
- When coding the LEVEL clause, it must immediately follow the VERSION clause. If a VERSION clause has not been entered, the LEVEL clause must immediately follow the initial action clause. If you specify a LEVEL, however, you cannot enter a THROUGH clause for the action.

All other clauses (following THROUGH, VERSION, and/or LEVEL) can be coded in any order.

Note: You can enter VERSION and LEVEL for the following actions, although this is not indicated in the syntax: ADD, ARCHIVE, GENERATE, MOVE, RESTORE, SIGNIN, TRANSFER, and UPDATE. However, these fields are ignored during processing.

SCL Execution JCL

As mentioned previously, you can use batch panels to enter your element action requests. The standard JCL required for execution is already defined. You most likely do not need to code additional JCL, except in special situations (for example, when you reference a file by DDname).

If you decide not to use the batch panels, you must code specific JCL in order to execute your requests. A sample of the JCL required is provided on the installation tape and loaded to the JCL library during installation. The JCL is located in *iprfx.iqual.*CSIQJCL member BC1JSCL.

&&ACTION Statement

The &&ACTION statement allows you to substitute any action for a specified element at run time. This statement normally is generated when you use the LIST action.

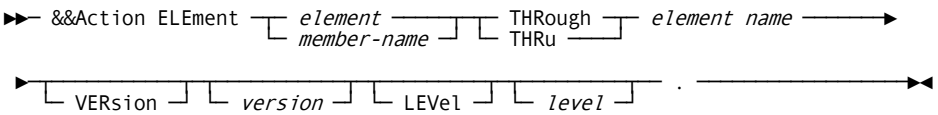
If you do not indicate a specific action(s) to be performed when you request a list, &&ACTION appears at the beginning of each clause. You can then input the appropriate action(s) at a later date, using the SET ACTION statement.

For example, at the beginning of a month, you may want to see a list of elements involved with a particular project, although you may not know what actions you will request for those elements. If you request the list without indicating any specific actions, &&ACTION appears, in lieu of a specific action, for every action card generated. When you are ready to perform individual actions, simply specify those actions with the necessary SET ACTION clause(s).

Whenever you execute an &&ACTION statement, you must precede it with a SET ACTION statement that contains the action to be performed. Depending on the action specified, you may need to include supplementary information, such as TO or FROM clauses, in the related SET ACTION statement. For more information, see the chapter "Using Set, Clear, and EOF Statements (see page 49)" and in The List Statement (see page 122).

Note: Additional clauses may be required depending on the action coded in the SET ACTION statement. Similarly, additional optional clauses will be available depending on the action you use. See the individual action descriptions for detailed information regarding each action's requirements and options.

The &&ACTION statement has the following syntax:



&&ACTION ELEMENTS element

member-name

Indicates the elements involved when the designated action is performed. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name. If you specify a level (in the LEVEL clause), however, you cannot use a name-mask with the element name.

THROUGH (THRU) element-name

Indicates the range of elements affected by the &&ACTION statement. You can use a name-mask with the element name. You cannot have both a THROUGH clause and a LEVEL clause.

VERSION version

Indicates the version you want to see for the specified element. Acceptable values are **1-99**.

You must code a full element name if you want to indicate a version number.

If you code the VERSION clause, it must follow the THROUGH clause.

LEVEL level

Indicates the level you want to see for the specified element. Acceptable values are **00-99**. By default, CA Endevor SCM retrieves the current level of the element.

If you enter a LEVEL clause, you cannot enter a THROUGH clause. In addition, you must code a full element name in the &&ACTION ELEMENTS clause.

The LEVEL option is not available for all actions. Check the individual action to see if this clause can be used.

Example: &&Action SCL

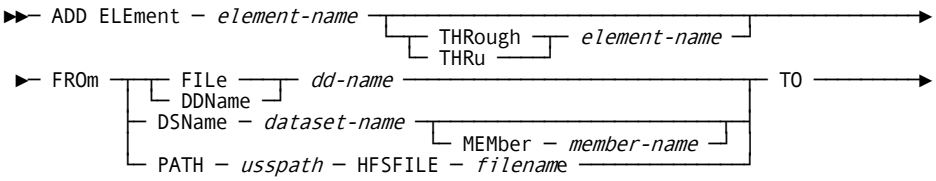
In this example, the SET ACTION GENERATE statement has been specified, as well the appropriate SET OPTIONS and SET FROM statements.

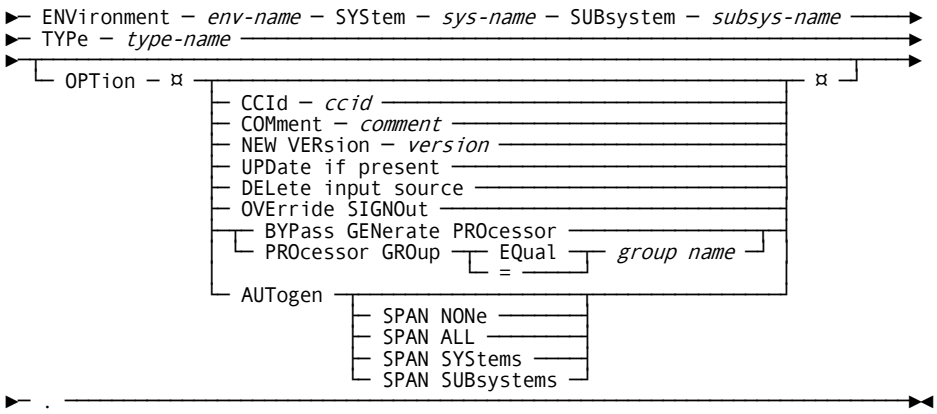
```
SET ACTION GENERATE .
SET FROM ENVIRONMENT 'PROD'
                            SYSTEM 'PAYROLL'
                            SUBSYSTEM 'REPORTS'
                            TYPE 'COBOL'
                            STAGE NUMBER 1 .
SET OPTIONS CCID REQ#43023
                            COMMENT 'REGENERATE WITH NEW COPY BOOKS'
                            COPYBACK
                            SEARCH .

&&ACTION ELEMENTS PAYRPT* .
```

Add Statement

The ADD statement allows you to add an element to an environment's entry stage in CA Endevor SCM. The entry stage for an environment is defined through the C1DEFLTS table.

The Add statement has the following syntax:





ADD ELEMENT element-name

Indicates the element or elements to be added. You can use a name-mask in the element name to specify more than one element. For more information about syntax rules, see Element Name Syntax Rules (see page 38).

THROUGH (THRU) element-name

Indicates that a range of elements should be added, beginning with the element coded in the ADD ELEMENTS statement, up to and including the element specified in this statement. You can use a name-mask with the element name. If you use the THROUGH clause, however, you cannot enter a member name in the FROM clause.

Note: If you are working with a sequential file, the THROUGH clause is ignored.

FROM FILE (DDNAME) dd-name

DSNAME dataset-name

MEMBER member-name

PATH usspath

HFSFILE filename

The FROM clause indicates the location of the element being added. CA Endevor SCM uses both the FROM clause in the action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

Note: For more information, see the description of the SET FROM statement in the chapter "Using Set, Clear, and EOF Statements (see page 49)."

The SET FROM statement allows you to specify only a file (DDname) or data set name, not a member name.

You must enter a FILE, DDNAME, DSNAME, or PATH in conjunction with HFSFILE (enter one and only one). If you enter either a FILE or DDNAME, be sure the appropriate JCL DD statement is coded.

Enter a member name if it differs from the element name specified in the ADD ELEMENTS clause. If you do not enter a member name, CA Endevor SCM assumes that the element name and member name are the same. If you provide a member name:

- The ADD ELEMENTS clause must contain a fully qualified element name.
- You cannot also code a THROUGH clause.

PATH

The USS directory where the element source file resides.

HFSFILE

The file in the USS directory that you want to put under the control of CA Endevor SCM.

TO **ENVIRONMENT env-name**

SYSTEM sys-name

SUBSYSTEM subsys-name

TYPE type-name

The TO clause indicates where the element is being added. CA Endevor SCM uses both the TO clause in the action and any preceding SET TO clause to determine the "to" criteria for that action.

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.

Note: For more information, see the description of the SET TO statement in the chapter "Using Set, Clear, and EOF Statements (see page 49)."

You must specify an environment, system, subsystem, and type for the ADD action. Remember that you cannot use a name-mask with any field in the TO location.

OPTIONS

OPTION clauses allow you to further specify action requests.

CCID *ccid*/COMMENT *comment*

You can enter a 1- to 12-character CCID and/or a 1- to 40-character comment.

CCIDs and/or comments may be required. If you do not provide a required CCID and/or comment, the ADD action fails.

When you specify a CCID and/or comment in an ADD action, CA Endevor SCM treats the CCID and/or COMMENT fields differently depending on whether you are adding a new element or an existing element.

- When you specify a CCID and/or comment in an ADD action for a new element, CA Endevor SCM uses this CCID and/or comment to:
  - Set the source and source delta CCID and/or COMMENT fields.
  - Set the generate and component list delta CCID and/or COMMENT fields if the generate processor is run.
  - Set the last action CCID and/or COMMENT fields.
  - CA Endevor SCM also clears the environment's entry stage RETRIEVE CCID and/or COMMENT fields when you add a new element.
- When you specify a CCID and/or comment in an ADD action for an existing element, CA Endevor SCM uses this CCID and/or comment to:
  - Set the source CCID and/or COMMENT fields if the CCID and/or comment has changed.
  - Set the source delta CCID and/or COMMENT fields.
  - Set the generate CCID and/or COMMENT fields if the generate processor is run.
  - Set the component list delta CCID and/or COMMENT fields if running the generate processor creates a change.
  - Set the last action CCID and/or COMMENT fields.

CA Endevor SCM also clears the environment's entry stage RETRIEVE CCID and/or COMMENT fields when you add an existing element. If you use the BYPASS GENERATE PROCESSOR option, the ADD action does *not* set the generate or component list delta CCID and/or COMMENT fields.

NEW VERSION version

If the element exists up the map, the version number associated with the existing element will be assigned, by default. If the New Version clause is omitted and the element does not exist up the map, the element is assigned version 1. If the New Version clause is used and the element exists at the target location or at a location up the mapped route, the ADD action fails.

UPDATE IF PRESENT

To successfully add an element to CA Endevor SCM, that element cannot currently exist in the entry stage.This also applies to a sourceless element. If the element is present in the entry stage, CA Endevor SCM returns an error message, regardless of whether the element at the entry stage is a sourced or sourceless element. The UPDATE IF PRESENT option, however, allows you to add the element even if it is in the entry stage, by automatically changing the ADD action to UPDATE.

DELETE INPUT SOURCE

After an element has been successfully added to CA Endevor SCM, you can use this option to remove the member from the library in which it originated.

If you input a sequential file, this option deletes that file.

OVERRIDE SIGNOUT

If the element has been signed out to a person other than yourself, you must code this option to perform this action. This option updates the signout ID at the appropriate stage, with the user ID of the person performing the override. Use OVERRIDE SIGNOUT with caution to avoid regressing changes made by another user.

BYPASS GENERATE PROCESSOR

Use this option if you do not want the generate processor executed for the element. Otherwise, as part of normal processing, CA Endevor SCM looks for and executes the generate processor for the element when it is added.

PROCESSOR GROUP EQ/= group name

You can specify that a particular processor group be used for this action. If you do not indicate a processor group and:

- You are adding a new member, the system defaults to the processor group associated with the type to which the element is assigned.
- You are adding an existing element, the system defaults to the processor group last used for this element.

AUTOGEN | SPAN NONE| SPAN ALL | SPAN SYSTEMS | SPAN SUBSYSTEMS

Applicable for Add, Update, and Generate actions in batch requests. This option cannot be used in packages and the option does not work if Bypass Generate Processor is set. The Global Type Sequencing batch processing method must be enabled. Autogen only acts on components whose Types are listed in the Global Type Sequencing table. If the component's Type is not listed in the Global Type Sequencing table, the Autogen request is ignored. In addition, your site must have purchased and activated the CA Endevor Automated Configuration. Autogen can be specified alone or with various Span keyword options. "Autogen Span" is *not* a valid option. Valid options follow:

AUTOGEN

Generates all elements that use the component that is the target of the action. These *using* elements are generated at the target location that is specified in the SCL statement. If they do not exist at the target location, they are brought back to the target location as sourceless elements. An administrator can change the behavior of the Autogen feature, by activating AUTOGEN_SOURCE in the Optional Features Table (ENCOPTBL). When this option is activated, the Generate actions for the using elements are built with the Copyback, instead of the NoSource, option. For more information about sourceless elements, see the NoSource option description in Generate Syntax (see page 116).

Note: *Using* elements are elements that use the element that is the target of an Add , Update, or Generate action. For example, if Autogen is specified for copybook, COPYA, then the programs that use that copybook are known as using elements.

AUTOGEN SPAN NONE

Generates all elements that use the component being acted upon. This option has the exact same effect as the option "AUTOGEN."

AUTOGEN SPAN ALL

Generates using elements that are found in any System and Subsystem combinations within the Environment and Stage of the component's logical map.

AUTOGEN SPAN SYSTEMS

Generates using elements found in any System, provided the element's Subsystem name matches the name of the Subsystem of the target component. Only Systems found within the Environment and Stage of the component's logical map or higher up the map are searched. This option is different from the Autogen option in that it includes additional Systems with the same Subsystem name in the search.

AUTOGEN SPAN SUBSYSTEMS

Generates using elements from all Subsystems with the same-named System of the component specified. Only Subsystem found in the System of the target component within the Environment and Stage of the component's logical map or higher up the map are searched. This option is different from the Autogen option in that it includes additional Subsystems with the same System in the search.

The following restrictions apply to the SPAN options:

- Common libraries must be included in the generate processor concatenations of the Systems and Subsystems of the using elements. If the libraries are not included, then the Generate action does include the changes made to the component element when searching across Systems or Subsystems.
- The same Systems and Subsystems must exist in each Environment and Stage location. For example, if ELEMENTA is a using element in the PROD Environment, system A, then system A must exist in the DEV Environment also.
- SPAN only includes using elements from Systems or Subsystems located in the target Environment and higher up the map.
- SPAN does not include using elements from outside the Environment map.

Example: Add SCL

This SCL adds a new element to the Payroll reporting subsystem in the environment PROD. After the ADD action completes, the source member will be deleted.

```
ADD ELEMENT 'PAYRPT31'
 TO  ENVIRONMENT 'PROD'
         SYSTEM 'PAYROLL'
     SUBSYSTEM 'REPORTS'
     TYPE ' COBOL'
 FROM DSNAME 'PAYROLL.SRCLIB'
 OPTIONS DELETE INPUT SOURCE
     CCID REQ#43213
     COMMENT 'ADD THE NEW PAYROLL REPORTING PROGRAM' .
```

Alter Statement

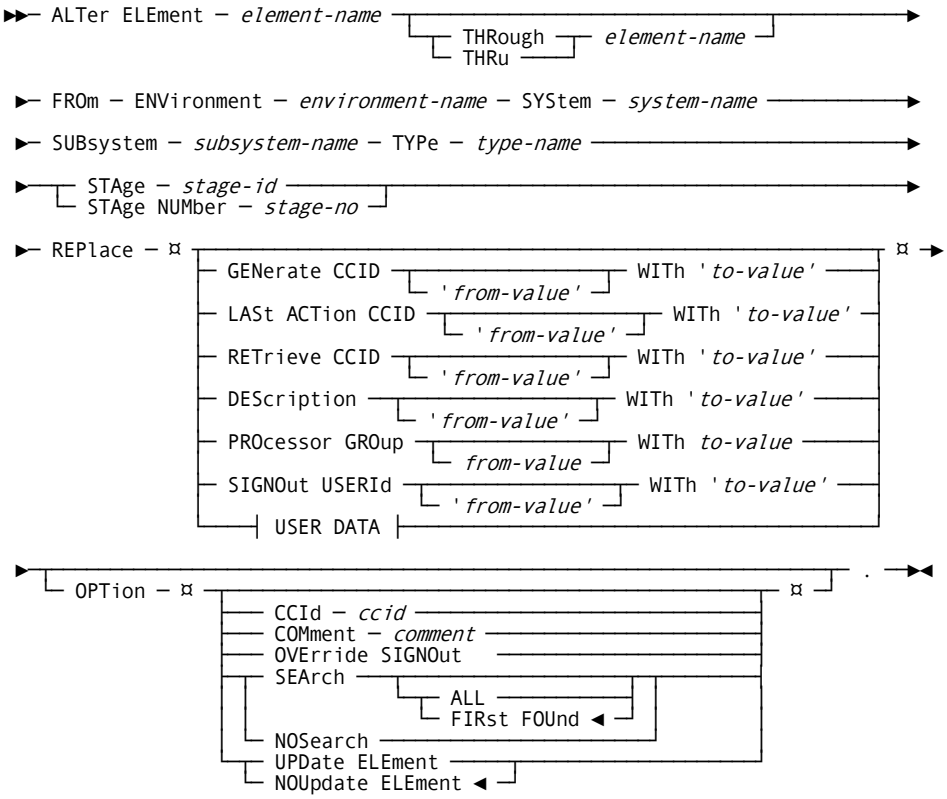
Important: The Alter action changes Master Control File metadata and cannot be easily reversed. Before you submit an Alter statement, we recommend that you back up the Master Control Files. If you wanted to undo changes, it might be easier to restore the Master Control Files. Otherwise, you would probably need to code multiple Alter statements to undo changes that were made using name-masked values.

Use the Alter statement to change element record metadata in the Master Control File. The Alter action alters the element metadata at the location where the matching elements are found. The action never fetches the element back to the specified From location.

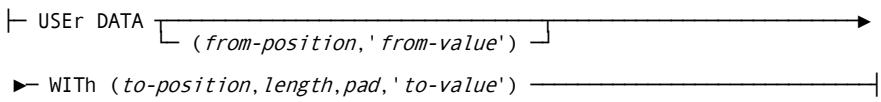
SMF recording must be active for any Environment in which you use the Alter action. If SMF recording is not active, the Alter action will fail. For more information about enabling SMF, see the *Administration Guide*.

Note: For more information about the Alter action, see the chapter "Altering Element Record Metadata" *in the Administration Guide*.

The Alter action statement has the following syntax:



Expansion of USER DATA:



Note: You can use name-masking to specify *some* variable values. A name-masked value is not explicit, because it includes a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. For more information about name-masking, see Name-Masking (see page 23).

Name-masking is allowed on all values except on the to-value, which is the new value that replaces the current value (the from-value) in the specified element record field. Therefore name-masking enables the same Alter statement to make the following changes:

- Changes can be made to multiple element instances. More than one element instance can be updated by one Alter statement, because name-masking on the element name and location enables Alter action processing to search for matches among multiple element instances.
- Changes can be made to different values. Name-masking on the field from-values (the current value) searches for all matching values. Omitting the current value or using a full wildcard (*) lets you change a field regardless of its current value.
- If name-masking is used on the element instance and on a current metadata field value, then multiple element instances can be updated to change multiple current field values.

You can change multiple metadata fields and multiple element instances in the same Alter statement as follows:

- The same Alter statement can specify changes to different metadata fields. For example, you can change the Last action CCID and the Description fields on the same statement.
- Each Alter statement can only specify one change to the same metadata field. For example, the Last action CCID field can only be specified once on the same statement.

Value specifications are case-sensitive, except for the Processor Group field, which is always in upper case.

The Alter action statement clauses are described next:

ALTER ELEMENT *element-name* THROUGH | THRU *element-name*

Changes the Master Control File metadata for the specified elements. You can name-mask the ELEMENT element-name. However, only the wildcard asterisk is supported on the THRU ELEMENT-NAME element-name. Long-named elements are supported.

FROM ENVIRONMENT *environment-name* SYSTEM *system-name* SUBSYSTEM *subsystem-name* TYPE *type-name* STAGE *stage-id* | STAGE NUMBER *stage-no*

Specifies the inventory location of the element that you want to alter. Specify an Environment, System, Subsystem, Type, and Stage or Stage Number. Name-masking is valid on all the variable values. The Search option searches for the element beginning with the specified Environment Stage and continues along the map. The Nosearch option limits the search to the specified Environment Stage. The Alter action alters the element metadata at the location where the matching elements are found and never fetches the element back to the specified From location.

REPLACE GENERATE CCID '*from-value*' WITH '*to-value*' LAST ACTION CCID '*from-value*' WITH '*to-value*' RETRIEVE CCID '*from-value*' WITH '*to-value*' DESCRIPTION '*from-value*' WITH '*to-value*' PROCESSOR GROUP '*from-value*' WITH '*to-value*' SIGNOUT USERID '*from-value*' WITH '*to-value*' USER DATA (*from-position*,'*from-value*') WITH (*to-position*, *length*, *pad*, '*to-value*')

Replaces the Master Control File metadata with the specified value. One or more options can be specified on the same Replace clause; they are *not* mutually exclusive. You can specify multiple Replace clauses on the same Alter statement. However, you cannot specify the same option more than once per Alter statement. All Replace from- and to-values are case-sensitive except for Processor Group. All Replace from- and to-values can contain embedded spaces if enclosed in quotes or double quotes, except for the Processor Group.

Whenever an Element metadata field is updated, an SMF record is written for the updated field. The Element metadata field is updated and the SMF record is written, even if the Element metadata field value is identical before and after the update.

From-value

The from-value variable specifies the current value in the Master Control File and can be name-masked or omitted.

- To replace a Master Control File field value regardless of the current field value, omit the from-value or specify the from-value using a full wildcard (*).
- To limit the field values that are replaced, specify an explicit or partly name-masked from-value. If the specified from-value matches the current field value, the field value is replaced.
- If the specified from-value does not match the current field value, the field value is not replaced and the action will have a return code of 4. However any matching replace fields on the request may still be performed. If none of the specified from-values matched and so the Master Control File record was not updated at all, then the action will have a return code of 8.

To-value

The to-value is required and specifies the new value. You can use name-mask characters (% and *), but they are treated as text values.

Specify one or more of the following options:

GENERATE CCID '*from-value*' WITH '*to-value*'

Specifies the current value and the replacement value for the 1- to 12-character CCID field, for the last Generate action in the Master Control File. The from-value supports name-masking, but the to-value treats name-mask characters as text characters. If the from-value is omitted or fully wildcarded, then the replacement will occur regardless of what is currently in the Master Control File field that is being interrogated. If the to-value string contains embedded spaces or you want to include leading spaces in the to-value string, it must be enclosed in quotes. This value is case-sensitive, regardless of the MIXEDFMT parameter setting in the C1DEFLTS table. You cannot alter the Generate CCID if the element was never Generated. Attempting to do so will cause the action to fail and none of the other replacements will be performed.

LAST ACTION CCID *'from-value'* WITH '*to-value*'

Specifies the current value and the replacement value for the 1- to 12-character CCID value, for the last action in the Master Control File. The from-value supports name-masking, but the to-value treats name-mask characters as text characters. If the from-value is omitted or fully wildcarded, then the replacement will occur regardless of what is currently in the Master Control File field that is being interrogated. If the to-value string contains embedded spaces or you want to include leading spaces in the to-value string, it must be enclosed in quotes. This value is case-sensitive, regardless of the MIXEDFMT parameter setting in the C1DEFLTS table.

RETRIEVE CCID '*from-value*' WITH *'to-value'*

Specifies the current value and the replacement value for the1- to 12-character CCID value, for the last Retrieve action in the Master Control File. The from-value supports name-masking, but the to-value treats name-mask characters as text characters. If the from-value is omitted or fully wildcarded, then the replacement will occur regardless of what is currently in the Master Control File field that is being interrogated. If the to-value string contains embedded spaces or you want to include leading spaces in the to-value string, it must be enclosed in quotes. This value is case-sensitive, regardless of the MIXEDFMT parameter setting in the C1DEFLTS table. You cannot alter the Retrieve CCID if the element was never Retrieved. Attempting to do so will cause the action to fail and none of the other replacements will be performed.

DESCRIPTION '*from-value*' WITH '*to-value*'

Specifies the element description to be replaced and the replacement value. Valid values can be 1 through 40 characters in length and are case-sensitive. The from-value supports name-masking, but the to-value treats name-mask characters as text characters. If the from-value is omitted or fully wildcarded, then the replacement will occur regardless of what is currently in the Master Control File field that is being interrogated. If the from-value or to-value string contains embedded spaces or you want to include leading spaces in the to-value string, it must be enclosed in quotes. To clear the field, enter a blank character in quotes. This value is case-sensitive, regardless of the MIXEDFMT parameter setting in the C1DEFLTS table.

Note: The description in the Master Control File was originally set by the comment field value coded when the element was first added. This value was also placed into the level 0 delta comment field. The to-value you code updates the Master Control File description field. The level 0 delta comment field is not modified.

PROCESSOR GROUP *from-value* WITH *to-value*

Specifies the name of the Processor Group to be replaced and the replacement value. Valid values can be one to eight national characters in length. The from-value supports name-masking, but name-mask characters are not valid on the to-value. If the from-value is omitted or fully wildcarded, then the replacement will occur regardless of what is currently in the Master Control File field that is being interrogated. The processor group specified on the to-value must exist, or the action will fail and none of the other replacements will be performed.

If *NOPROC* is coded on the from-value or to-value, the asterisk (*) characters are treated as ordinary characters in the text string instead of name-mask characters.

Note: When the Alter action changes a Processor Group, the action does not execute the Delete Processor on the old Processor Group or the Generate Processor on the new Processor Group. Performing a Generate action, at any time after the Alter was performed with a processor group change, will not necessarily remove any orphaned components or output libraries.

SIGNOUT USERID '*from-value*' WITH '*to-value*'

Specifies the Signout UserID to be replaced and the replacement value. Valid values can be one to eight national characters in length and are case-sensitive. Name-masking is supported on the from-value. If the from-value is omitted or fully wildcarded, then the replacement will occur regardless of what is currently in the Master Control File field that is being interrogated. No edits are performed on the Signout UserID field. It is up to the operating system to enforce naming standards.

USER DATA (*from-position*,'*from-value*') WITH (*to-position*, *length*, *pad*, '*to-value*')

Specifies what part of the current User Data field will be matched and where a new text string will replace all or part of the current User Data field. The new value overwrites all or part of the current value. Valid values can be 1 through 80 characters in length and are case-sensitive. Name-masking is supported on the from-value.

Note: User Data can also be changed through an exit 2 (before action) program on Add, Update, and Generate actions. When processing the Alter action, the exit 2 field that sets the user data (assembler field REQUSR or COBOL field REQ-USER-DATA) is ignored. Any Add, Update, or Generate actions that call an exit 2 program to set the user data field can overwrite a user data value previously set by the Alter action.

USER DATA (*from-position,'from-value*')

Specifies the value to be matched and where the search for that value begins.

- If from data is coded, it must be completely coded. The parentheses are required.

  from-position— Specifies the position within the User Data field to match against the from-value.

  from-value— Specifies the value to be matched. If the string contains embedded spaces, it must be enclosed in quotes. Name-masking is supported.

- The from data can be completely omitted. If no from data is coded, no search is performed and the new text string overwrites the specified position, regardless of the current value of the entire field. For example, to replace whatever is in columns 21 through 25 with TIME=, code the following: REPLACE USER DATA WITH (21,5,,'TIME=').

WITH (*to-position*, *length*, *pad*, '*to-value*')

Specifies the value to be added and where to add it. The length and pad values are optional, but either both must be used or both omitted. If the length and pad values are not used, the commas are still required, for example: WITH (to-position, , , 'to value’)

- To-position— Specifies where the replacement value begins.
- Length— Optional. Specifies the length of the replacement text. If the to-value is longer than the length variable, an error message is issued.
- Pad— Optional. Specifies the character that is used to pad the replacement string so that the length of the replacement string matches the length variable. If the to-value is shorter than the length variable, the pad character is required to lengthen the replacement text.
- To-value— Specifies the replacement value. If the string contains embedded spaces, it must be enclosed in quotes. Name-masking is not supported for the text string. If the * or % characters are coded, they are treated as text characters.

Examples of valid Replace clauses for User Data follow:

- REPLACE USER DATA WITH (1,5,A,'TASK')

  Replaces the text in columns 1 through 5 with TASKA.

- REPLACE USER DATA (1,ZZZ) WITH (25,5,A,'User')

  Replaces the text found in positions 25 through 29 with the value of UserA when the from-value found in position 1 is ZZZ.

- REPLACE USER DATA WITH (1,,,'USERA')

  Replaces the text beginning in column 1 with USERA.

OPTION CCID *ccid-value* COMMENT *comment-value* OVERRIDE SIGNOUT UPDATE | NOUPDATE SEARCH [ALL | FIRST FOUND] | NOSEARCH

Specifies processing options.

CCID *ccid-value*

Specifies a 1- to 12-character CCID to be added to the SMF record for the target element. For the Alter action, this option does not update the Master Control File. This value may be case-sensitive depending on the MIXEDFMT parameter setting in the C1DEFLTS table.

COMMENT *comment-value*

Specifies a 1- to 40-character comment to be added to the SMF record for the target element. For the Alter action, this option does not update the Master Control File. This value may be case-sensitive depending on the MIXEDFMT parameter setting in the C1DEFLTS table.

OVERRIDE SIGNOUT

Specifies that you want to update the element, even if the element is signed out to someone else. If the element is signed out to someone else, you must code this option to perform the Alter action. However, the Alter action retains the current Signout Userid for the element; it does not update the Signout UserID with the user ID of the person performing the Alter action. If you want the Signout Userid element field to be changed as part of the Alter action, use the Replace Signout UserID clause.

SEARCH [ALL | FIRST FOUND] | NOSEARCH

Specifies a logical or physical search of the software inventory that is based on the C1DEFLTS table Environment definitions. After an Environment Stage is matched, the Stage Master Control File is searched based on other criteria: the Alter action element name specification and the Replace and Options clauses.

SEARCH [ALL | FIRST FOUND]

Specifies a logical search of the software inventory that is based on the mapping of Environment Stages as defined in the C1DEFLTS table. The search begins with the Environment Stage that is specified on the element action From clause and continues along the map. On the From clause, the Environment name must be explicit. Name-masking is supported on the Stage ID. If the Stage ID is name-masked, the starting location is Stage 1, regardless of whether that Stage is in the logical map or is the entry Stage for the Environment. The Alter action changes the Master Control File for each matching element at the location where the element is found.

The ALL and First Found options are ignored by all actions, except the Alter action.

ALL— Identifies all elements that are found in the logical search. If the Alter action is run in Update mode, then all the elements that are found in the search get updated.

FIRST FOUND— Identifies the first element that is found in the logical search. If the Alter action is run in Update mode, then only the first element that is found in the search gets updated. The default.

NOSEARCH

Specifies a physical search of the software inventory that is based on an exact match to the C1DEFLTS table Environment definitions, regardless of mapping. The search is limited to the Environment Stage that is specified on the element action From clause. Name-masking is supported on the Environment name (for the ALTER action only) and on the Stage ID.

If the Environment or Stage is name-masked, more than one Environment Stage can meet the selection criteria. For example, If you specified an environment-name of D* and the C1DEFLTS table includes the unmapped environments DEV, QA, PRD, and DEMO, then both the DEV and DEMO Environments would match.

UPDATE | NOUPDATE ELEMENT

Specifies the action mode. Specify one of the following modes:

UPDATE ELEMENT— Updates the Master Control File.

NOUPDATE ELEMENT— Identifies the elements that meet the Alter selection criteria, but does not update the Master Control File. The default.

Note: The UPDATE | NOUPDATE ELEMENT option cannot be coded in the SET statement.

Example— Alter Action Replacement of User Data

The following examples show User Data replacement:

- If the following is coded, the to-length is 80, the replacement text length is 16, the to-position is 1, and the pad character is a space.

```
ALTER ELEMENT TESTELM1 OPTION UPDATE ELEMENT REPLACE
    USER DATA      (1,'ENV1')  WITH (1,80,' ','ENV1234 SYSTEM01')
```

If the original user data field contained this:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
ENV1    SYSTEM99SUBSYS99TYPE0099                                   TESTUPDATEACT
```

It would contain this after the replacement:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
ENV1234 SYSTEM01
```

- If the following is coded, the to-length is 80, the replacement text length is 13, the to-position is 1, and the pad character is @.

```
ALTER ELEMENT TESTELM1 OPTION UPDATE ELEMENT REPLACE
    USER DATA      (1,'ENV1')  WITH (1,80,'@','ENV1 SYSTEM01')
```

If the original user data field contained this:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
ENV1    SYSTEM99SUBSYS99TYPE0099                                   TESTUPDATEACT
```

It would contain this after the replacement:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
ENV1 SYSTEM01@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

- If the following is coded, the to-length is 25, the replacement text length is 8, the to-position is 15, and the pad character is @.

```
ALTER ELEMENT TESTELM1 OPTION UPDATE ELEMENT REPLACE
USER DATA (1,'ENV1') WITH (15,25,'@','88SYSSUB')
If the original user data field contained this:
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+--
--8
ENV1    SYSTEM99SUBSYS99TYPE0099                                TESTUPDATEACT

It would contain this after the replacement:
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+--
--8
ENV1    SYSTEM88SYSSUB@@@@@@@@@@@@@@@@@@                        TESTUPDATEACT
```

Archive Statement

The ARCHIVE statement writes the base level and all change levels of an element to a sequential file (known as an archive data set). In addition, for CA Endevor SCM ACM users, the ARCHIVE action writes the base level and all change levels of the Component List to the archive data set.

Use the ARCHIVE action to:

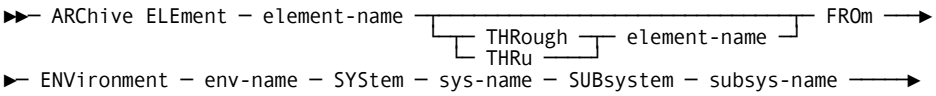
- Maintain a backup copy of the element source.
- Delete the existing version of a particular element from Stage 2.
- Maintain an archive version of the element source, that is, to maintain a version of the element source outside of CA Endevor SCM.

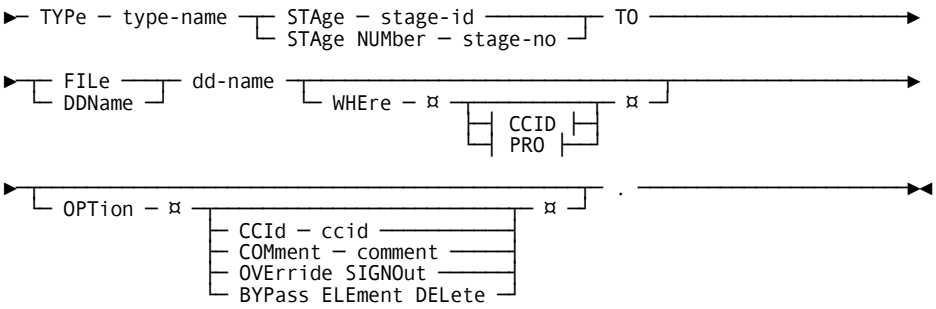
Archive is available in batch only. Once an element has been archived, COPY, LIST, RESTORE, and TRANSFER actions can be executed against the archive data set.

Sourceless elements cannot be archived. When an attempt to archive an explicitly specified sourceless element is encountered, the action fails. If a sourceless element is encountered during wildcard expansion, the element is skipped and a warning message is issued.

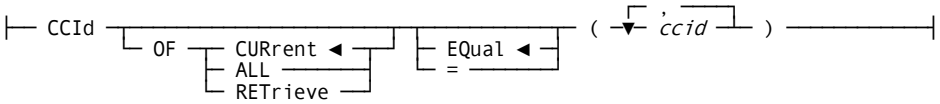
When the Bypass Element Delete option is not specified, a check is made to see if prior sourceless elements exist. If one does exist, the action will fail if no other sourced element exists upstream. The action cannot be allowed to continue because it would remove source access to the sourceless element, making it impossible for it to be regenerated.

The Archive statement has the following syntax:

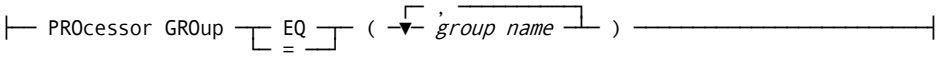




Expansion of CCID



Expansion of PRO



ARCHIVE ELEMENTS element-name

Indicates the element(s) to be archived. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name.

THROUGH (THRU) element-name

Indicates that a range of elements should be archived, beginning with the element coded in the ARCHIVE ELEMENTS statement, up to and including the element specified in this statement. You can use a name-mask with the element name.

FROM ENVIRONMENT env-name

SYSTEM sys-name

SUBSYSTEM subsys-name

TYPE type-name

STAGE stage-id

STAGE NUMBER stage-no

The FROM clause indicates the location of the element being archived. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

You must specify an environment, system, subsystem, type, and stage. The environment name must be explicit. You can use a name-mask with the system, subsystem, type, and stage. The stage specification can be either one of the following:

- STAGE ID-Enter a single alphanumeric stage identifier.
- STAGE NUMBER-Enter either **1** or **2**.

TO FILE (DDNAME) dd-name

The TO clause indicates where the element is being archived. CA Endevor SCM uses both the TO clause in an action and any preceding SET TO clause to determine the "to" criteria for that action.

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.

The DCB must specify variable blocked records (RECFM=VB), and the DSORG should be PS.

- The minimum LRECL should be 2940 or the TYPE-LENGTH plus (+) 14, whichever is greater.
- When archiving to disk, the recommended block size is one-half a track, and the recommended LRECL is one-half a track minus (-) 4 unless the previous rule requires a bigger LRECL/BLKSIZE.
- When archiving to tape, the recommended block size is 32760 and the recommended LRECL is 32756.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID OF *ccid*

Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

- **CURRENT-**Tells CA Endevor SCM to look through the CCID fields in the MCF (Master Control File) to find a specified CCID(s). This is the default.
- **ALL-**Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s).
- **RETRIEVE-**Tells CA Endevor SCM to use the CCID in the Master Control File RETRIEVE CCID field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary. The next examples illustrate the use of this clause.

```
Example 1: WHERE CCID OF CURRENT (PROJ001, PROJ002, PROJ004)
Example 2: WHERE CCID OF ALL (PROJ00V)
```

WHERE PROCESSOR GROUP EQ/= *group name*

This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1: WHERE PROCESSOR GROUP (COBVS, COBII)
Example 2: WHERE PROCESSOR GROUP (COBV)
```

OPTIONS

OPTIONS clauses allow you to further specify requests.

CCID *ccid*/COMMENT*comment*

You can enter a 1- to 12- character CCID and/or a 1- to 40-character comment.

CCIDS and/or comments may be required. If you do not provide a required CCID and/or comment, the ARCHIVE action fails.

This is the CCID that CA Endevor SCM looks for if WHERE ARCHIVE CCID is specified for the LIST, COPY, RESTORE, and TRANSFER actions.

OVERRIDE SIGNOUT

If the element has been signed out to a person other than yourself, you must code this option in order to perform this action. Note, however, that OVERRIDE SIGNOUT does not apply when you select the BYPASS ELEMENT DELETE option for this action. This option updates the SIGNOUT ID at the appropriate stage with the user ID of the person performing the override. Use OVERRIDE SIGNOUT with caution to avoid regressing changes made by another user.

BYPASS ELEMENT DELETE

Use this option if you do not want the element automatically deleted (the default) after it is archived. Otherwise, CA Endevor SCM deletes the element, that is, the base and all change levels.

Example: Archive SCL

This SCL archives all of the elements from the Payroll Reporting subsystem. The archived elements will be written to the preallocated DD name "ARCHOUT.' The signout status will be overridden, if necessary.

```
ARCHIVE ELEMENT '*'
   FROM ENVIRONMENT 'PROD'
                SYSTEM 'PAYROLL'
                SUBSYSTEM 'REPORTS'
                TYPE 'COBOL'
                STAGE NUMBER 1
  TO DDNAME ARCHOUT
  OPTIONS CCID REQ#44145
                COMMENT 'ARCHIVE REPORTING SUBSYSTEM PROGRAMS'
                OVERRIDE SIGNOUT .
```

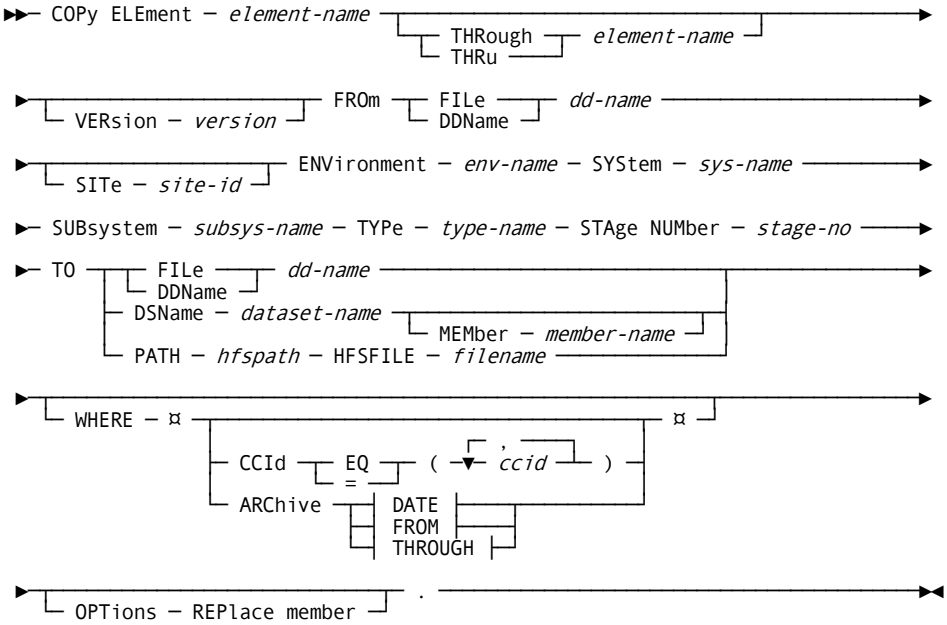
Copy Statement

The COPY statement copies an element from an archive data set to a user data set (that is, a data set external to CA Endevor SCM). The user data set can be a library (a CA Panvalet file, a CA Librarian file, or a PDS) or a sequential file. The element is not restored to the Master Control File.

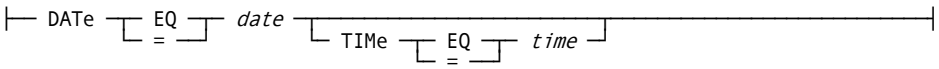
The COPY action is available in batch only.

Note: Copy processing is strictly external to CA Endevor SCM.

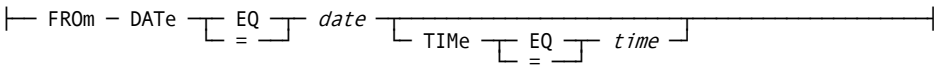
The Copy statement has the following syntax:



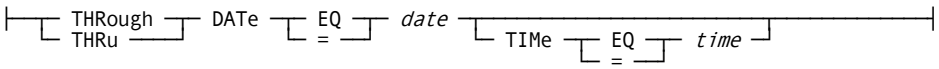
Expansion of DATE



Expansion of FROM



Expansion of THROUGH



COPY ELEMENTS *element-name*

Indicates the element(s) you want to copy. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name.

THROUGH (THRU) *element-name*

Indicates that you want to copy a range of elements, beginning with the element coded in the COPY ELEMENTS statement, up to and including the element specified in this statement. You can use a name-mask with the element name. If you enter the THROUGH clause, however, you cannot enter a member name (in the TO clause).

VERSION *version*

Indicates the version of the element you want to copy. Acceptable values are **1-99**.

You must code a full element name if you want to indicate a version number.

If you code the VERSION clause, it must follow the THROUGH clause.

FROM FILE (DDNAME) *dd-name*

ENVIRONMENT *env-name*

SYSTEM *sys-name*

SUBSYSTEM *subsys-name*

TYPE *type-name*

STAGE NUMBER *stage-no*

The FROM clause indicates the location of the element being copied. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

The FILE (DDNAME) portion of the clause is required. The file name indicates from which archive file the element is being copied. Enter this information first when coding the syntax.

You must specify an environment, system, subsystem, type, and stage number (either **1** or **2**). The environment name must be explicit. You can use a name-mask with the system, subsystem, type, and stage.

Entering a site ID is optional. This field further defines the location of the element being copied.

TO **FILE (DDNAME)** ***dd-name***

DSNAME ***dataset-name***

MEMBER ***member-name***

The TO clause indicates the file or data set name to which the element is being copied. CA Endevor SCM uses both the TO clause in an action and any preceding SET TO clause to determine the "to" criteria for that action.

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.

You must enter either a FILE, a DDNAME, or a DSNAME (enter one and only one). If you enter a FILE or DDNAME, be sure the appropriate JCL is coded.

Enter a member name if it differs from the element name specified in the COPY ELEMENTS clause. Remember that you cannot use a name-mask with a TO field name.

If you do not enter a member name, CA Endevor SCM assumes that the element name and member name are the same.

- You can enter a member name only if a full element name has been coded in the COPY ELEMENTS clause; that is, if you have not used a name-mask.
- If you want to code a member name, you must do so in the COPY statement; the SET TO MEMBER clause does not apply to the COPY action. If you do enter a member name, you cannot use the THROUGH clause.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID EQ/= ***ccid***

Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1: WHERE CCID EQ PROJ00V
Example 2: WHERE CCID  (PROJ001, PROJ002, PROJ004)
```

WHERE ARCHIVE

This clause allows you to select elements based on the date and, optionally, time that an element was archived. There are four possible forms for this clause:

WHERE ARCHIVE DATE mm/dd/yy [TIME hh:mm]

This clause tells CA Endevor SCM to copy only those elements with this date, and optionally, time stamp.

WHERE ARCHIVE FROM DATE mm/dd/yy [TIME hh:mm]

This clause tells CA Endevor SCM to copy all elements with a date and, optionally, time stamp on or after the specified date and time stamps.

WHERE ARCHIVE THROUGH DATE mm/dd/yy [TIME hh:mm]

This clause tells CA Endevor SCM to copy all elements with a date and, optionally, time stamp earlier than and including the specified date and time stamp.

WHERE ARCHIVE FROM DATE mm/dd/yy [TIME hh:mm] TH ROUGH DATE mm/dd/yy [TIME hh:mm]

This clause tells CA Endevor SCM to copy only those elements with a date, and optionally, time stamps within the specified range. If you enter a time, you must enter the date with it.

OPTIONS REPLACE MEMBER

If the element you are copying exists in the target location, CA Endevor SCM rejects the request unless you code the REPLACE MEMBER option. Specify this option when you want to replace the existing member in the library.

Example: Copy SCL

This SCL copies the archived version of Payroll program "PAYRPT43" to a user data set. The input is taken from a DDname that refers to a data set that was created with the ARCHIVE action.

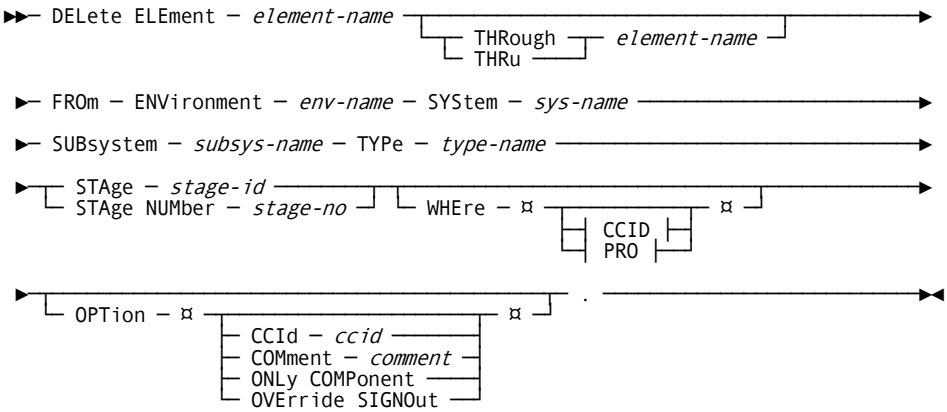
```
COPY ELEMENT 'PAYRPT43'
                FROM DDNAME ARCHIVE
                           ENVIRONMENT 'PROD'
                           SYSTEM 'PAYROLL'
                           SUBSYSTEM 'REPORTS'
                           TYPE 'COBOL'
                           STAGE NUMBER 1
                TO DSNAME 'PAYROLL.SRCLIB' MEMBER 'PAYRPT43'
                OPTIONS REPLACE MEMBER .
```

Delete Statement

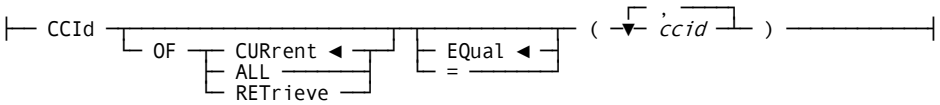
The DELETE statement deletes an element from the specified inventory location.

When deleting a sourced element, if a prior sourceless element exists and if no other sourced element exists upstream from the delete location, the action will fail. The action cannot be allowed if there is no sourced instance of the element upstream from a sourceless element. This would make a regeneration of a sourceless element impossible.

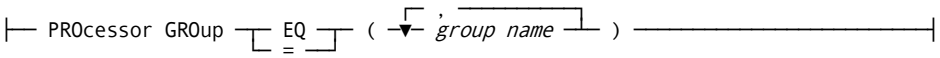
The Delete statement has the following syntax:



Expansion of CCID



Expansion of PRO



DELETE ELEMENTS element-name

Indicates the element(s) to be deleted. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name.

THROUGH (THRU) element-name

Indicates that a range of elements should be deleted, beginning with the element coded in the DELETE ELEMENTS statement, up to and including the element specified in this statement. You can use a name-mask with the element name.

FROM ENVIRONMENT env-name

SYSTEM sys-name

SUBSYSTEM subsys-name

TYPE type-name

STAGE stage-id

STAGE NUMBER stage-no

The FROM clause indicates the location of the element being deleted. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

You must specify an environment, system, subsystem, type, and stage. The environment name must be explicit. You can use a name-mask with the system, subsystem, type, and stage. The stage specification can be either one of the following:

- STAGE ID-Enter a single alphanumeric stage identifier.
- STAGE NUMBER-Enter either **1** or **2**.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID OF *ccid*

Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

CURRENT-Tells CA Endevor SCM to look through the CCID fields in the MCF (Master Control File) to find a specified CCID(s). This is the default.

ALL-Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s).

RETRIEVE-Tells CA Endevor SCM to use the CCID in the Master Control File RETRIEVE CCID field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1: WHERE CCID OF CURRENT (PROJ001, PROJ002, PROJ004)
Example 2: WHERE CCID OF ALL (PROJ00V)
```

WHERE PROCESSOR GROUP EQ/= *group name*

This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1: WHERE PROCESSOR GROUP (COBVS, COBII)
Example 2: WHERE PROCESSOR GROUP (COBV)
```

OPTIONS

OPTIONS clauses allow you to further specify action requests.

CCID ccid/COMMENT comment

You can enter a 1- to 12- character CCID and/or a 1- to 40-character comment.

CCIDs and/or comments may be required. If you do not provide a required CCID and/or comment, the DELETE action fails.

ONLY COMPONENTS

Applicable for CA Endevor SCM ACM users only. Indicates whether you want to delete both the element component list and the element, or the element component list only. Y (yes-delete just the element component list) or N (no-delete the element as well as the element component list).

OVERRIDE SIGNOUT

If the element has been signed out to a person other than yourself, you must code this option in order to perform this action. This option updates the SIGNOUT ID at the appropriate stage with the user ID of the person performing the override. Use OVERRIDE SIGNOUT with caution to avoid regressing changes made by another user.

Example: Delete SCL

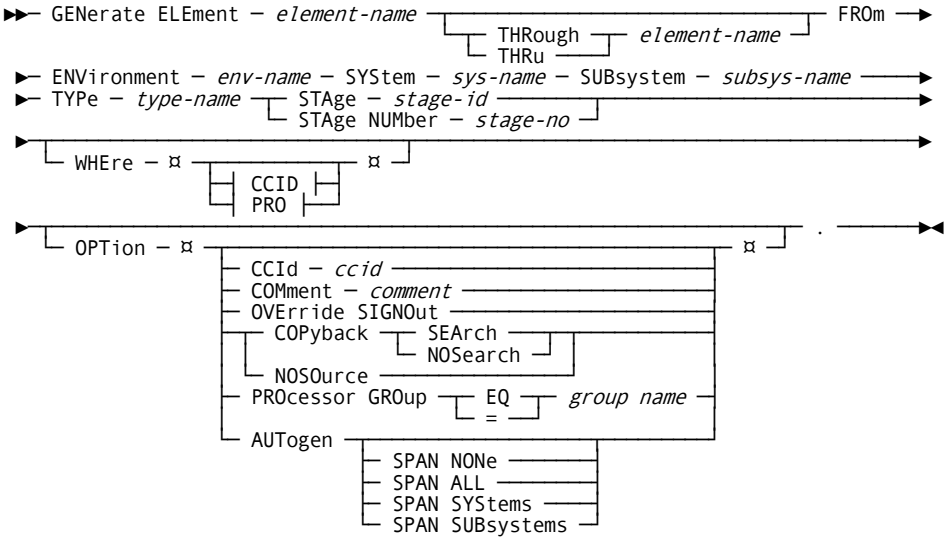
This SCL deletes an element from Stage 2. The signout will be overridden, if necessary.

```
DELETE ELEMENT 'PAYRPT03'
   FROM ENVIRONMENT 'PROD'
        SYSTEM 'PAYROLL'
        SUBSYSTEM 'REPORTS'
        TYPE 'COBOL'
        STAGE NUMBER 2
        OPTIONS CCID REQ#43034
        COMMENT 'DELETE AN OBSOLETE PAYROLL PROGRAM'
        OVERRIDE SIGNOUT .
```

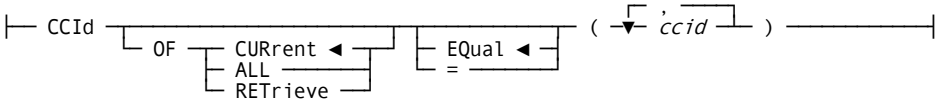
Generate Statement

The GENERATE statement executes the generate processor for the current level of an element, in either Stage 1 or Stage 2.

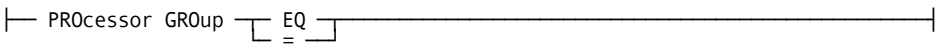
The Generate statement has the following syntax:



Expansion of CCID



Expansion of PRO



GENERATE ELEMENTS element-name

Indicates the element(s) to be generated. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name.

THROUGH (THRU) element-name

Indicates that a range of elements should be generated, beginning with the element coded in the GENERATE ELEMENTS statement, up to and including the element specified in this statement. You can use a name-mask with the element name.

FROM ENVIRONMENT env-name
SYSTEM system-name
SUBSYSTEM subsys-name
TYPE type-name
STAGE stage-id
STAGE NUMBER stage-no

The FROM clause indicates the location of the element being generated. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

Note: For more information, see the description of the SET FROM statement in the chapter "Using Set Clear, and EOF Statements (see page 49)."

You must specify an environment, system, subsystem, type, and stage. The environment name must be explicit. You can use a name-mask with the system, subsystem, type, and stage. The stage specification can be either one of the following:

- STAGE ID-Enter a single alphanumeric stage identifier.
- STAGE NUMBER-Enter either **1** or **2.**

If you use a name-mask, CA Endevor SCM begins searching for the specified element(s) in Stage 1 of the current environment, and generates the first element that matches the specified element name, regardless of its location, version or level.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.

If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID OF *ccid*

Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

CURRENT

Tells CA Endevor SCM to look through the CCID fields in the MCF (Master Control File) to find a specified CCID(s). This is the default.

ALL

Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s).

RETRIEVE

Tells CA Endevor SCM to use the CCID in the Master Control File's RETRIEVE CCID field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1: WHERE CCID OF CURRENT (PROJ001, PROJ002, PROJ004)
Example 2: WHERE CCID OF ALL (PROJ00V)
```

WHERE PROCESSOR GROUP EQ/= *group name*

This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1: WHERE PROCESSOR GROUP (COBVS, COBII)
Example 2: WHERE PROCESSOR GROUP (COBV)
```

OPTIONS

OPTIONS clauses allow to further specify an action request.

CCID ccid/COMMENT comment

You can enter a 1- to 12- character CCID and/or a 1- to 40-character comment.

CCIDs and/or comments may be required. If you do not provide a required CCID and/or comment, the GENERATE action fails.

When you specify a CCID and/or comment in a GENERATE action, CA Endevor SCM updates the CCID and/or COMMENT fields differently, depending on whether you specify the GENERATE action with or without the COPYBACK option.

When you specify a CCID and/or comment in a GENERATE action without the COPYBACK option, CA Endevor SCM uses this CCID and/or comment to:

- Set the generate CCID and/or COMMENT fields.
- Set the last action CCID and/or COMMENT fields.
- Set the component list delta CCID and/or COMMENT fields if running the generate processor creates a change.

When you specify a CCID and/or comment in a GENERATE action with the COPYBACK option, CA Endevor SCM uses this CCID and/or comment to:

- Set the generate and component list delta CCID and/or COMMENT fields.
- Set the last action CCID and/or COMMENT fields.

CA Endevor SCM also uses the CCID and comment associated with the copied-back element to set the source and source delta CCID and/or COMMENT fields when you generate that element using the COPYBACK option.

COPYBACK

If you select this option, CA Endevor SCM first copies the current level of the element back to the FROM stage, then generates the element. CA Endevor SCM searches for the element first in the current environment, then in other stages along the map.

If the element currently exists in the FROM stage, CA Endevor SCM ignores the COPYBACK option and simply generates the element.

COPYBACK cannot be used with NOSOURCE.

SEARCH or NOSEARCH

This option is valid only when you have selected the COPYBACK option. The SEARCH option tells CA Endevor SCM to look for the element to be generated with copyback along the map, if it is not in the current environment.

Code NOSEARCH to restrict the search to the current environment.

NOSOURCE

When the target location has a sourced element, the element is generated in place.

When the target location has a sourceless element, the element is generated at the target location using the source of the first occurance of the element found up the map.

When the element does not exist at the target location, the element is generated at the target location using the source of the first occurance of the element found up the map. The source is not fetched to the target. The MCF element created at the target location will contain data similiar to a fetched back element except that the element base and delta name fields will be blank and the record will be marked as a sourceless element.

NOSOURCE cannot be used with COPYBACK. It is not necessary to specify the SEARCH option with NOSOURCE, because NOSOURCE implies SEARCH.

OVERRIDE SIGNOUT

If the element has been signed out to a person other than yourself, you must code this option in order to perform this action. This option updates the SIGNOUT ID at the appropriate stage with the user ID of the person performing the override. Use OVERRIDE SIGNOUT with caution to avoid regressing changes made by another user.

PROCESSOR GROUP EQ/= *group name*

Select this option to specify a predefined, named group of processors. If you do not specify a processor group, CA Endevor SCM defaults to the processor group last used for this element.

AUTOGEN | SPAN NONE| SPAN ALL | SPAN SYSTEMS | SPAN SUBSYSTEMS

Applicable for Add, Update, and Generate actions in batch requests. This option cannot be used in packages and the option does not work if Bypass Generate Processor is set. The Global Type Sequencing batch processing method must be enabled. Autogen only acts on components whose Types are listed in the Global Type Sequencing table. If the component's Type is not listed in the Global Type Sequencing table, the Autogen request is ignored. In addition, your site must have purchased and activated the CA Endevor Automated Configuration. Autogen can be specified alone or with various Span keyword options. "Autogen Span" is *not* a valid option. Valid options follow:

AUTOGEN

Generates all elements that use the component that is the target of the action. These *using* elements are generated at the target location that is specified in the SCL statement. If they do not exist at the target location, they are brought back to the target location as sourceless elements. An administrator can change the behavior of the Autogen feature, by activating AUTOGEN_SOURCE in the Optional Features Table (ENCOPTBL). When this option is activated, the Generate actions for the using elements are built with the Copyback, instead of the NoSource, option. For more information about sourceless elements, see the NoSource option description in Generate Syntax (see page 116).

Note: *Using* elements are elements that use the element that is the target of an Add , Update, or Generate action. For example, if Autogen is specified for copybook, COPYA, then the programs that use that copybook are known as using elements.

AUTOGEN SPAN NONE

Generates all elements that use the component being acted upon. This option has the exact same effect as the option "AUTOGEN."

AUTOGEN SPAN ALL

Generates using elements that are found in any System and Subsystem combinations within the Environment and Stage of the component's logical map.

AUTOGEN SPAN SYSTEMS

Generates using elements found in any System, provided the element's Subsystem name matches the name of the Subsystem of the target component. Only Systems found within the Environment and Stage of the component's logical map or higher up the map are searched. This option is different from the Autogen option in that it includes additional Systems with the same Subsystem name in the search.

AUTOGEN SPAN SUBSYSTEMS

Generates using elements from all Subsystems with the same-named System of the component specified. Only Subsystem found in the System of the target component within the Environment and Stage of the component's logical map or higher up the map are searched. This option is different from the Autogen option in that it includes additional Subsystems with the same System in the search.

The following restrictions apply to the SPAN options:

- Common libraries must be included in the generate processor concatenations of the Systems and Subsystems of the using elements. If the libraries are not included, then the Generate action does include the changes made to the component element when searching across Systems or Subsystems.
- The same Systems and Subsystems must exist in each Environment and Stage location. For example, if ELEMENTA is a using element in the PROD Environment, system A, then system A must exist in the DEV Environment also.
- SPAN only includes using elements from Systems or Subsystems located in the target Environment and higher up the map.
- SPAN does not include using elements from outside the Environment map.

Example: Generate SCL

This SCL generates COBOL program PAYRPT01 at Stage 1. The element will be fetched if it does not already exist at Stage 1.

```
GENERATE ELEMENT 'PAYRPT17'
   FROM ENVIRONMENT 'PROD'
                SYSTEM 'PAYROLL'
                SUBSYSTEM 'REPORTS'
                TYPE COBOL
                STAGE NUMBER 1
  OPTIONS CCID REQ#43023
                COMMENT 'REGENERATE WITH NEW COPY BOOKS'
                COPYBACK
                SEARCH .
```

List Statement

The LIST statement scans elements or members in the Master Control File, an archive data set, or a library, and generates a list of elements/members that meet your selection criteria. The LIST action is available in batch only. The WHERE clause supplies the selection criteria for the LIST action. It selects the elements based on content as opposed to the names of the elements.

The LIST action searches for elements and/or members in a location defined by the data you enter in the FROM clause. You can request a LIST action from one of the following:

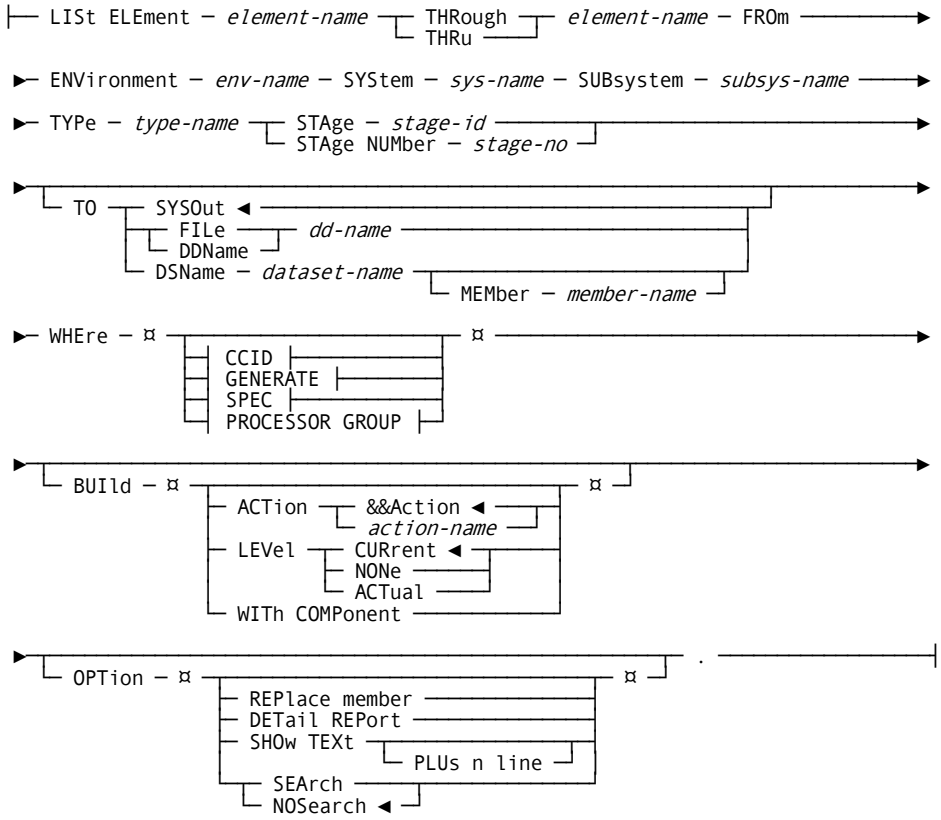
- CA Endevor SCM (Master Control File)
- An archive data set
- An external library

The processing involved is the same for each type of LIST request. The clauses required, however, depend on the location being searched. Similarly, the options available depend on the location of the element or member. This section of the chapter addresses each type of LIST request separately; the appropriate syntax is illustrated first, followed by a complete discussion of the associated LIST action rules.

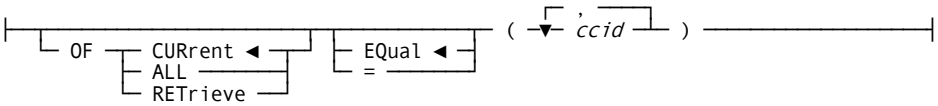
List From CA Endevor SCM Statement

The LIST FROM CA Endevor SCM statement generates a list of elements from CA Endevor SCM's Master Control File.

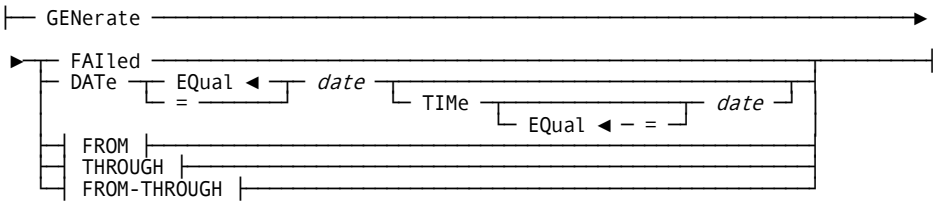
List From CA Endevor SCM Syntax



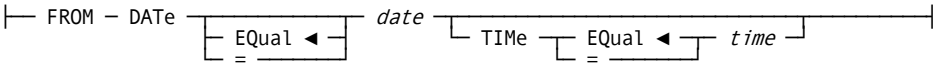
Expansion of CCId



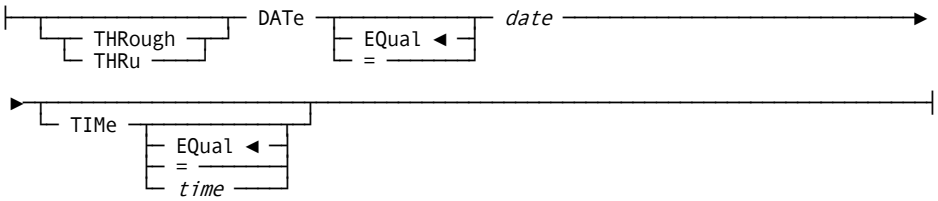
Expansion of GENERATE



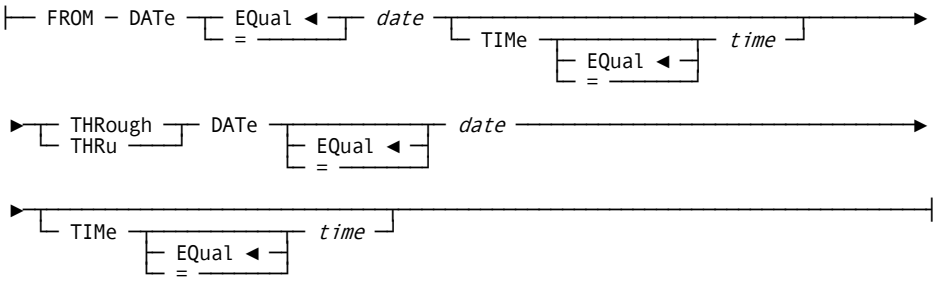
Expansion of FROM



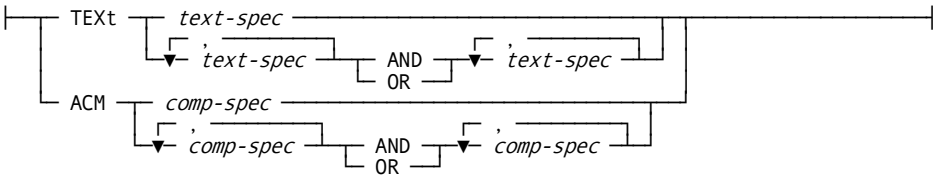
Expansion of THRough



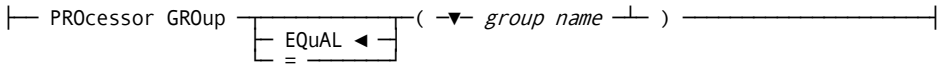
Expansion of FROM-THROUGH



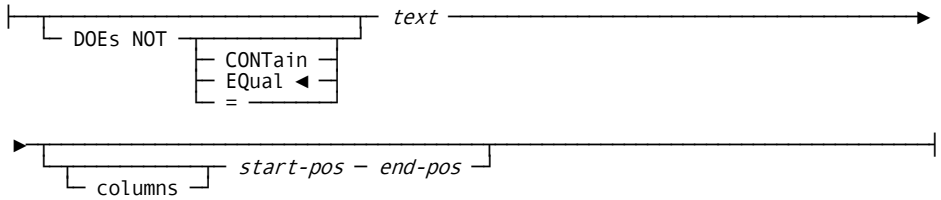
Expansion of SPEC



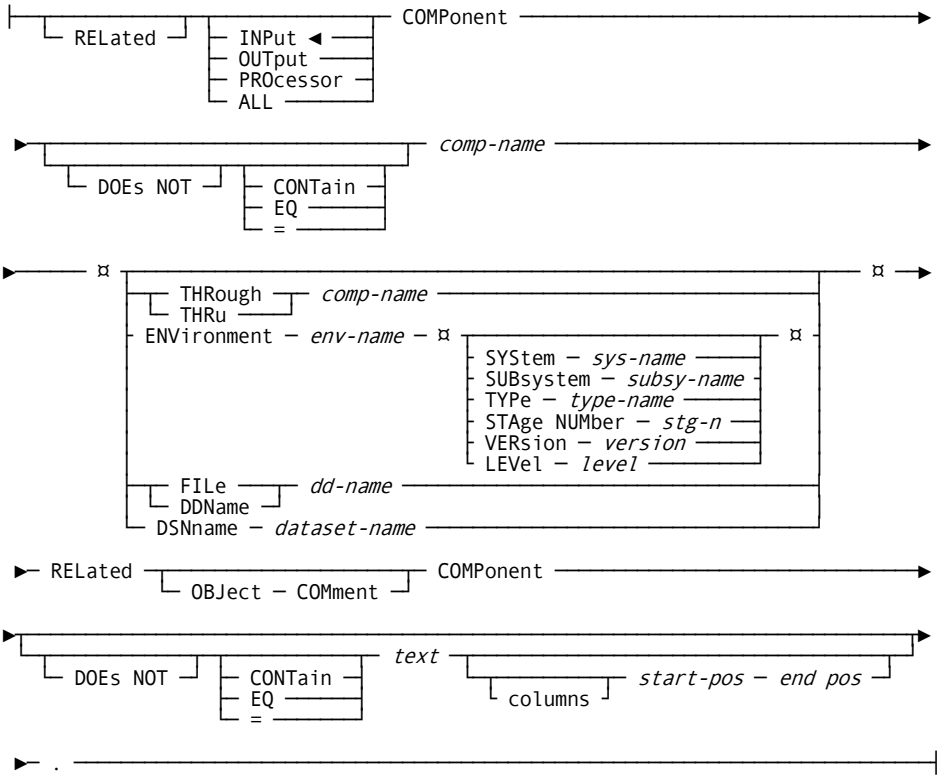
Expansion of PROCESSOR GROUP



Where text-spec is replaced as necessary with:



Where comp-spec is replaced as necessary with:



LIST ELEMENT *element-name*

Indicates the element(s) to be listed. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name.

THROUGH (THRU) *element-name*

Indicates that a range of elements should be listed, beginning with the element coded in the LIST ELEMENT clause, up to and including the element specified in this statement. You can use a name-mask with the element name. If you code the THROUGH clause, you cannot enter a member name (in the TO clause).

FROM ENVIRONMENT *env-name*
SYSTEM *system-name*
SUBSYSTEM *subs-name*
TYPE *type-name*
STAGE *stage-id*
STAGE NUMBER *stage-no*

The FROM clause indicates the location of the element to be listed. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

You must specify an environment, system, subsystem, type, and stage. The environment name must be explicit. You can use a name-mask with the system, subsystem, type, and stage. The stage specification can be either one of the following:

- STAGE ID-Enter a single alphanumeric stage identifier.
- STAGE NUMBER-Enter either **1** or **2**.

TO

The TO clause indicates where the element is to be listed. CA Endevor SCM uses both the TO clause in an action and any preceding SET TO clause to determine the "to" criteria for that action.

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.

TO SYSOUT-

SYSOUT is the default TO location. If you do not provide a TO clause, and no SET TO information has been coded, CA Endevor SCM writes the action cards to the Execution Report. The Execution Report appears immediately after the LIST request and cannot be edited.

TO FILE (DDNAME) *dd-name* or DSNAME *dataset- name*

You can tell CA Endevor SCM to write action cards to both the Execution Report and an external data set by providing a file name (DDname) or a data set name; if you enter a FILE or DDNAME, be sure the appropriate JCL is coded. Use this option if you want to edit the action cards.

TO MEMBER *member-name*

Enter a member name. This clause is valid only if you are not specifying a sequential file.

- CA Endevor SCM ignores a member specification if you have coded the TO SYSOUT option.
- The action fails if you code a member name along with a file (DDname) or data set name that is sequential.

If you are using PDSs and do not provide a member name, CA Endevor SCM assigns a temporary name of TEMPNAME. If you wish to use the temporary naming capability, do not code multiple list requests to the same external data set.

If you are using a PDS and have multiple list statements with only one member name on a SET statement, then all lists go to same member name and only first LIST results are available.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID OF *ccid*

Limits the list to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

- **CURRENT-**Tells CA Endevor SCM to look through the CCID fields in the MCF (Master Control File) to find a specified CCID(s). This is the default.
- **ALL-**Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s). If the element is sourceless, the source element delta level CCID checks are not done, because the element source and delta files are not present.
- **RETRIEVE-**Tells CA Endevor SCM to use the CCID in the Master Control File RETRIEVE CCID field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1: WHERE CCID OF CURRENT (PROJ001, PROJ002, PROJ004)
Example 2: WHERE CCID OF ALL (PROJ00V)
```

WHERE GENERATE

This clause allows you to select elements based on the date and, optionally, time that an element was generated. There are five forms for this clause.

WHERE GENERATE FAILED-

Tells CA Endevor SCM to list only those elements for which the generate processor failed.

WHERE GENERATE DATE *mm/dd/yy* [TIME *hh:mm*]-

Tells CA Endevor SCM to list only those elements with this date, and optionally, time stamp.

WHERE GENERATE FROM DATE *mm/dd/yy* [TIME *hh:mm*] -

Tells CA Endevor SCM to list all elements with a date and, optionally, time stamp on or after the specified date and time stamps.

WHERE GENERATE THROUGH DATE *mm/dd/yy* [TIME *hh:mm*]-

Tells CA Endevor SCM to list all elements with a date and, optionally, time stamp earlier than and including the specified date and time stamp.

WHERE GENERATE FROM DATE *mm/dd/yy* [TIME *hh:mm*] THROUGH DATE *mm/dd/yy* [TIME *hh:mm*]-

Tells CA Endevor SCM to list only those elements with date, and optionally, time stamps within the specified range. If you enter a time, you must enter the date with it.

WHERE TEXT *text spec*-

Limits the list to elements that contain (or do not contain) one or more specified 1- to 70-character text strings. All “WHERE TEXT” filters are bypassed on sourceless elements. Examples follow.

- In this example, CA Endevor SCM lists all elements containing the text string WO9-LINKAGE:

  ```
  WHERE TEXT 'WO9-LINKAGE'
  ```

- In this example, CA Endevor SCM lists all elements containing the text strings COPY COPY005 and COPY COPY010 between columns 8 and 40 of the element source:

  ```
  WHERE TEXT ((EQ 'COPY COPY005' COLUMN 8 40) AND EQ 'COPY COPY010' COLUMN 8 40))
  ```

- In this example, CA Endevor SCM lists all elements that do not contain the text string REMARKS between columns 8 and 15 of the element source:

  ```
  WHERE TEXT DOES NOT CONTAIN 'REMARKS' COLUMN 8 15
  ```

- In this example, CA Endevor SCM lists all elements that contain either the text string M605SUB or the text string M607SUB and do not contain the text string M606SUB:

  ```
  WHERE TEXT (('M605SUB' OR 'M607SUB') AND DOES NOT CONTAIN 'M606SUB')
  ```

The WHERE TEXT EQUAL clause cannot be used with the WHERE ACM clauses.

WHERE [ACM] *comp spec*-

Limits the list to component lists containing the designated component name. Wildcards are acceptable in the component name specification.

WHERE INPUT COMPONENT

This is the default. It tells CA Endevor SCM to list both input components and related input components matching your entry.

WHERE RELATED INPUT COMPONENT

Tells CA Endevor SCM to list only related input components matching your entry.

WHERE OUTPUT COMPONENT

Tells CA Endevor SCM to list both output components and related output components matching your criteria.

WHERE RELATED OUTPUT COMPONENT

Tells CA Endevor SCM to list only related output components matching your entry.

WHERE PROCESSOR COMPONENT

Tells CA Endevor SCM to list only processor components matching the criteria.

WHERE ALL COMPONENT

Tells CA Endevor SCM to list matches within all three types of components.

WHERE RELATED OBJECT COMPONENT-

Tells CA Endevor SCM to list only objects matching your entry.

WHERE RELATED COMMENT COMPONENT

Tells CA Endevor SCM to list comments matching your entry. Note that this applies only to comments that have been added to the component list by the CONRELE utility in a processor step. For more information on the CONRELE utility, see the Extended Processors Guide.

You can further specify the component using the following clauses.

THROUGH (THRU) ***comp-name***

Tells CA Endevor SCM to list elements containing one in a specific range of component names. The range begins with the component name coded in the WHERE COMPONENTS clause, and encompasses all components up to and including the component specified in this clause. Wildcards are acceptable in the component name specification.

ENVIRONMENT ***env name***

Tells CA Endevor SCM to list elements with components located in the specified environment. If you provide an environment name you must also provide the following:

SYSTEM-1 to 8 characters
SUBSYSTEM-1 to 8 characters
TYPE-1 to 8 characters
STAGE NUMBER-either **1** or **2**

VERSION ***version***

Tells CA Endevor SCM to list elements containing components with a specific version number. Acceptable values are 00-99. The version number of the component may differ from the version number of the element with which it is associated.

LEVEL ***level-***

Tells CA Endevor SCM to list elements containing components with a specific level number. Acceptable values are 00-99. The level number of the component may differ from the level number of the element with which it is associated.

FILE (DDNAME) ***dd-name-***

Tells CA Endevor SCM to list elements whose:

Input components originated from the specified DDname;

Output components were written to the specified DDname;

Components were produced by a processor step specified by and associated with the designated DDname.

DSNAME data set name-

Tells CA Endevor SCM to list elements whose:

Input components originated from the specified data set;

Output components were written to the specified data set;

Components were produced by a processor step specified by and associated with the designated data set.

WHERE ACM comp spec {AND/OR} comp spec

Allows you to provide compound component selection criteria. See the previous section for option descriptions. The WHERE ACM clauses cannot be used with the WHERE TEXT clause.

WHERE PROCESSOR GROUP group name

This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1:  WHERE PROCESSOR GROUP  (COBVS, COBII)
Example 2:  WHERE PROCESSOR GROUP  (COBV)
```

BUILD

Indicates specific information to be applied to each action statement generated by LIST. The data in this clause assists in building the SCL statements (and builds additional SCL statements if you enter WITH COMPONENTS) that result from your LIST request.

Note: You cannot build a generate with copyback request using LIST. This is because LIST can only build an action for an element when the element exists in the FROM stage.

If you do not enter BUILD information, CA Endevor SCM looks for a SET BUILD clause containing the appropriate information. If a SET BUILD clause has not been coded, the system defaults to &&ACTION for BUILD ACTION and to CURRENT for BUILD LEVEL.

The WITH COMPONENTS clause is optional within the BUILD clause; if it is not coded here or in the SET BUILD statement, component information is not provided in the list.

BUILD ACTION

Determines the action that appears in the LIST action syntax for the specified element. You can enter either a specific action name or &&ACTION, which indicates that a specific action will be designated for this element at a later time. This action can be entered manually or using the SET ACTION statement.

BUILD LEVEL

Indicates whether you want the version and level of the specified element to appear on the action cards generated by the LIST request:

CURRENT-Tells CA Endevor SCM to include the current version and level of elements in LIST actions. This is the default if the WHERE COMPONENTS SPEC clause has not been coded for the action, or no component list exists (CA Endevor SCM ACM is not installed). If the where component spec clause has been coded for the action, the default is ACTUAL.

NONE-Tells CA Endevor SCM not to list the current version and level for the element.

ACTUAL-When building using ACM information, tells CA Endevor SCM to include the level of the component as recorded in the component list in LIST action statements, rather than the current level of the element as recorded in the Master Control File. This is the default if a WHERE COMPONENTS SPEC clause has been coded for the action.

When building from source level, create action statements using the VERSION and LEVEL of the source level that matches the WHERE clause.

BUILD WITH COMPONENTS

Indicates that action cards should be generated for every input component that is associated with the specified element. BUILD WITH COMPONENTS pertains to the CA Endevor SCM ACM product only, and must be used in conjunction with the WHERE ACM clause (explained earlier in this section).

OPTIONS

OPTIONS clauses allow you to further specify action requests.

REPLACE MEMBER

When you specify a PDS and member name in the TO clause, list requests fail if the member already exists. Use the REPLACE MEMBER option if you want to replace the existing member in the TO location library.

DETAIL REPORT

By default, in the Execution Report, CA Endevor SCM lists only those elements matching the selection criteria you specify. If you select the DETAIL REPORT option, every element searched is listed in the report-whether or not a match is found.

SHOW TEXT [PLUS n LINES]

This option allows you to print the line of source code that contains a specified text string, plus a designated number of lines of code before and after the text string.

You must code the WHERE TEXT clause if you use the SHOW TEXT option. Otherwise, you receive a syntax error.

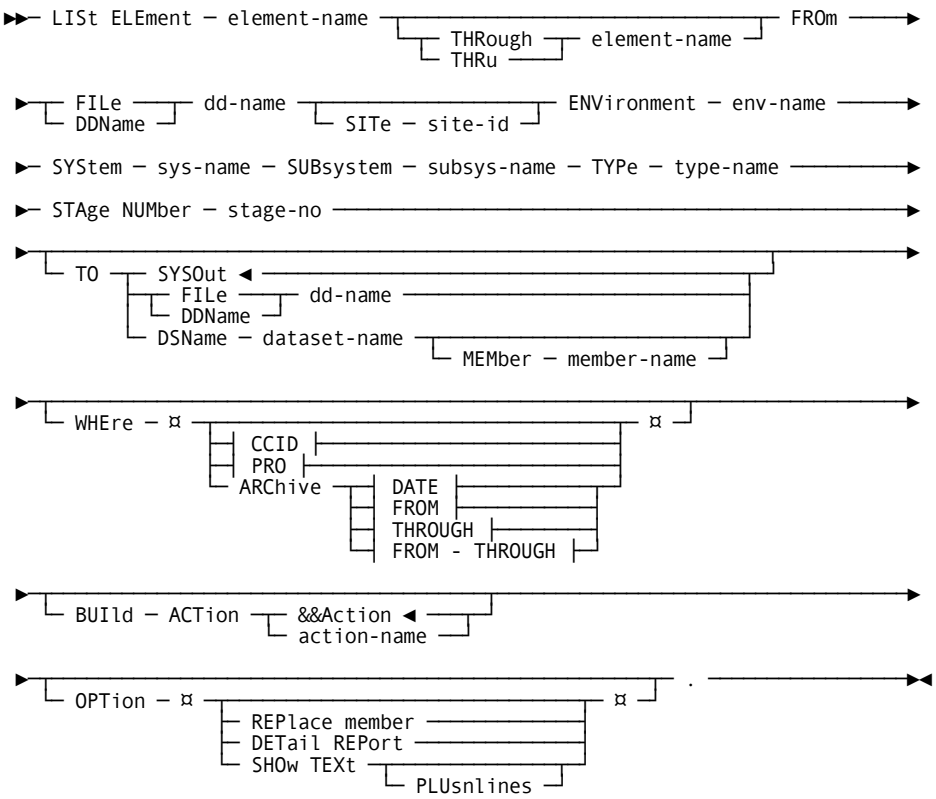
SEARCH or NOSEARCH

The SEARCH option tells CA Endevor SCM to look for and list all occurrences of the element on the map. The default is NOSEARCH. Code NOSEARCH to restrict CA Endevor SCM's list to the current environment.

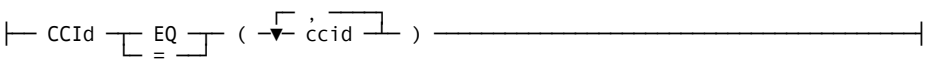
List from Archive Data Set

The LIST FROM ARCHIVE DATA SET statement generates a list of elements from an archive data set.

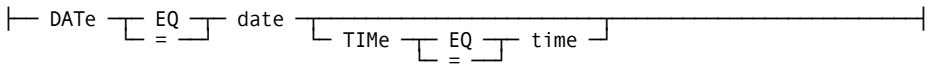
List from Archive Data Set Syntax



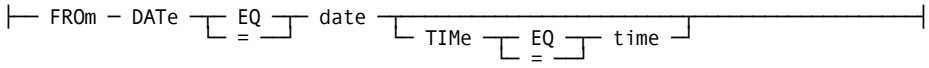
Expansion of CCID



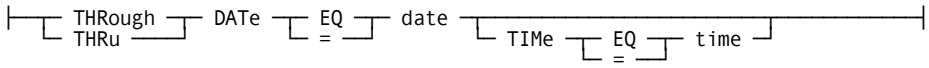
Expansion of DATE



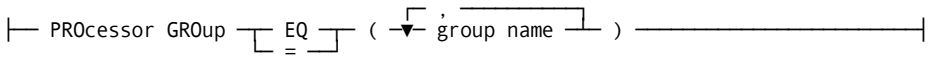
Expansion of FROM



Expansion of THROUGH



Expansion of PRO



LIST ELEMENT *element-name*

Specifies the elements to be listed. You can specify a name-masked element name.

THROUGH (THRU) *element-name*

Specifies a range of elements to be listed, beginning with the element coded in the LIST ELEMENT clause, up to and including the element specified in this clause. You can specify a name-masked element name. If you code the THROUGH clause, you cannot enter a member name in the TO clause.

FROM FILE (DDNAME)*dd-name* SITE *site-id*
ENVIRONMENT *env-name*
SYSTEM *system-name*
SUBSYSTEM *subsys-name*
TYPE *type-name*
STAGE NUMBER *stage-no*

Specifies the location of the element to be listed. You must code a FILE or DDNAME for the archive data set to be searched for the specified element. Enter this information first when coding the syntax. The SITE site-id is optional.You must specify an environment, system, subsystem, type, and stage number (either 1 or 2). The environment name must be explicit. You can name-mask the system, subsystem, type, and stage.

The From clause and any preceding Set From statements determine the from criteria for the action. Specify the from criteria according to the following rules:

- The from criteria must be fully specified, except for the SITE site-id. This includes the archive file ddname and all the inventory location specifications (Environment, System, Subsystem, Stage, Type, and Stage).
- All the From fields can be specified on a single Set From statement, without the existence of the FROM clause or any FROM fields within the action syntax.
- All the From fields can be specified within the action syntax, without the existence of a Set From statement.
- To be valid, a Set From statement must specify the archive FILE ddname and one or more of the inventory location specifications, and no other Set From statements can exist before the action statement.
- If the Set From statement is coded correctly, any of the FROM inventory location field values can be overridden in the action syntax. The FROM clause on the action must specify any of the criteria missing from the Set From statement.

In the following cases, an action with an invalid from criteria specification can complete processing, but list elements from the production inventory, instead of the archive data set:

- If more than one Set From statement exists. In this case, only the last statement is used, but the location defaults to the production environment.
- If any of the criteria are missing, the location defaults to the production environment.

Note: If the action executes against the archive file, the following message appears in the Action Execution Report:

```
C1G0213I        FROM ARCHIVE FILE: ARCHFILE
```

If the action executed, but you do not see this message, the action was performed against the production inventory location that is coded on the request. If the elements exist in the production environment, the action may complete successfully. However, the list will include elements found in the production environment, instead of the archive data set. If the elements do *not* exist in the production environment, a message is issued that indicates that the ENV env-name is not found.

Note: For more information about the from criteria and the SET FROM statement, see Examples: From Criteria Specification for List Element From Archive (see page 138).

TO

Specifies where the element is to be listed. CA Endevor SCM uses both the TO clause in an action and any preceding SET TO clause to determine the "to" criteria for that action.

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.

TO SYSOUT-

SYSOUT is the default TO location. If you do not provide a TO clause, and no SET TO information has been coded, CA Endevor SCM writes the action cards to the Execution Report. The Execution Report appears immediately after the LIST action request and cannot be edited.

TO FILE (DDNAME) dd-name or DSNAME dataset-name-

You can tell CA Endevor SCM to write action cards to both the Execution Report and an external data set by providing a file name (DDname) or a data set name; if you enter a FILE or DDNAME, be sure the appropriate JCL is coded. Use this option if you want to edit the action cards.

TO MEMBER member-name-

Enter a member name. This clause is valid only if you are **not** specifying a sequential file.

CA Endevor SCM ignores a member specification if you have coded the TO SYSOUT option.

- The action fails if you code a member name along with a file (DDname) or data set name that is sequential.
- If you are using PDSs and do not provide a member name, CA Endevor SCM assigns a temporary name of TEMPNAME. If you wish to use the temporary naming capability, do not code multiple list requests to the same external data set.

If you are using a PDS and have multiple list statements with only one member name on a SET statement, then all lists go to same member name and only first LIST results are available.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID ccid -

Limits the list to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1:  WHERE CCID EQ PROJ00V
Example 2:  WHERE CCID (PROJ001, PROJ002, PROJ004)
```

WHERE ARCHIVE-

This clause allows you to select elements based on the date and, optionally, time that an element was archived. There are four possible forms for this clause:

WHERE ARCHIVE DATE mm/dd/yy [TIME hh:mm]

This clause tells CA Endevor SCM to list only those elements with this archive date, and optionally, time stamp.

WHERE ARCHIVE FROM DATE mm/dd/yy [TIME hh:mm]

This clause tells CA Endevor SCM to list all elements with an archive date and, optionally, time stamp on or after the specified date and time stamps.

WHERE ARCHIVE THROUGH DATE mm/dd/yy [TIME hh:mm]

This clause tells CA Endevor SCM to list all elements with an archive date and, optionally, time stamp earlier than and including the specified date and time stamp.

WHERE ARCHIVE FROM DATE mm/dd/yy [TIME hh:mm] THROUGH DATE mm/dd/yy [TIME hh:mm]

This clause tells CA Endevor SCM to list only those elements with archive date, and optionally, time stamps within the specified range. If you enter a time, you must enter the date with it.

WHERE PROCESSOR GROUP *group name*-

This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1:  WHERE PROCESSOR GROUP  (COBVS, COBII)
Example 2:  WHERE PROCESSOR GROUP  (COBV)
```

BUILD ACTION&&ACTION | *action-name*

Determines the action that appears in the LIST action syntax for the specified element. If you do not enter required BUILD information here, CA Endevor SCM looks for a SET BUILD clause containing the appropriate information. If a SET BUILD clause has not been coded, the system defaults to &&ACTION.

You can enter a specific action (for example, ADD or MOVE) in this clause or the variable &&ACTION. &&ACTION indicates that a specific action will be designated for this element at a later time. This action can be entered manually or using the SET ACTION statement.

OPTIONS

OPTIONS clauses allow you to further specify action requests.

REPLACE MEMBER-When you specify a PDS and member name in the TO clause, LIST requests fail if the member already exists. Use the REPLACE MEMBER option if you want to replace the existing member in the TO location library.

DETAIL REPORT-By default, in the Execution Report, CA Endevor SCM lists only those elements matching the selection criteria you specify. If you select the DETAIL REPORT option, every element searched is listed in the report-whether or not a match is found.

SHOW TEXT [PLUS n LINES]-This option allows you to print the line of source code that contains a specified text string, plus a designated number of lines of code before and after the text string.

Examples: From Criteria on List Element from Archive Action

The following examples show valid and invalid from criteria specifications for the List Element From Archive statement.

These examples work as expected to perform the action against the archive location:

- All the FROM fields can be coded on a single SET statement, as shown next.

```
SET FROM FILE
    ARCHFILE
    ENV ENV1
    SYS 'P7015'
    SUB 'P7015'
    TYP 'P7015'
    STA NUM 1.
LIST ELEMENT *
    TO DDNAME SCLSTMTS MEMBER LIST1
BUILD ACTION &&ACTION
 .
```

- All the FROM fields can be coded within the action syntax without the existence of a SET FROM statement containing inventory location information, as shown next:

```
SET STOPRC 16 .
LIST ELEMENT *
    FROM FILE
    ARCHFILE
    ENV 'ENV1' SYSTEM 'P7015' SUBSYSTEM 'P7015' TYPE 'P7015' STA NUM 1
    TO DDNAME SCLSTMTS MEMBER LIST2
BUILD ACTION &&ACTION
 .
```

- When mixing the FROM fields between a SET statement and the action, the SET FROM FILE and at least one of the FROM inventory location fields must be specified in a single SET statement and no other SET statement containing FROM inventory location information can exist. A valid example is shown next:

```
SET FROM FILE
    ARCHFILE
    ENV ENV1
    TYP 'P7015'
    STA NUM 1.
LIST ELEMENT *
FROM SYSTEM 'P7015' SUBSYSTEM P7015
    TO DDNAME SCLSTMTS MEMBER LIST4
BUILD ACTION RESTORE
 .
```

- When mixing the FROM fields between a SET statement and the action, the SET FROM FILE and at least one of the FROM inventory location fields must be specified in a single SET statement and no other SET statement containing FROM inventory location information can exist. Overrides in the action syntax are supported. A valid example is shown next:

```
SET FROM FILE
    ARCHFILE
    ENV ENVZ
    TYP 'ZZZZ'
    STA NUM 1.
LIST ELEMENT *
FROM ENV ENV1 SYSTEM 'P7015' SUBSYSTEM P7015 TYPE P7015 STA NUM 2
    TO DDNAME SCLSTMTS MEMBER LIST4
BUILD ACTION TRANSFER
 .
```

- In the following example the BUILD ACTION value is DELETE. This is supported, but the FROM FILE ddname and WHERE ARCHIVE DATE SCL clauses are suppressed in the generated SCL so they can be executed against an CA Endevor SCM Environment instead of an archive file. DELETE is the only action that works this way.

```
SET FROM FILE
    ARCHFILE
    SUBSYSTEM 'P7015' .
LIST ELEMENT *
FROM ENV 'ENV1' SYSTEM P7015 SUBSYSTEM P7015 TYPE 'P7015'  STA NUM 1
    TO DDNAME SCLSTMTS MEMBER DELETE
BUILD ACTION DELETE.
 .
```

  Generated SCL statements;

```
SET FROM ENVIRONMENT  ENV1      SYSTEM  P7015     SUBSYSTEM  P7015
         TYPE  P7015     STAGE NUMBER 1
.
DELETE   ELEMENT   P7015A        .
DELETE   ELEMENT   P7015B        .
DELETE   ELEMENT   P7015C        .
```

These examples are coded incorrectly and do not perform the action against the archive location:

- In the following example, more than one SET FROM statement exists. The second SET FROM clears the first and CA Endevor SCM assumes the list action is against the production inventory location instead of an ARCHIVE file. This action may execute successfully depending on whether elements exist at the production inventory location, but it is probably not what the request was intended to do. Message C1G0213I will not be found in the CA Endevor SCM Execution Report associated with this action.

```
SET FROM FILE
    ARCHFILE.
SET FROM
    SUBSYSTEM 'P7015' .
LIST ELEMENT *
FROM ENV 'ENV1' SYSTEM 'P7015' TYPE 'P7015' STA NUM 1
    TO DDNAME SCLSTMTS MEMBER DELETES
BUILD ACTION &&ACTION
 .
```

- In the following example the second SET FROM statement cancels out the first SET FROM statement. Therefore the SET FROM including the subsystem is ignored and the action fails due to the missing subsystem specification.

```
SET FROM FILE
    ARCHFILE
    SUBSYSTEM 'P7015' .
SET FROM
    SYSTEM    'P7015' .
LIST ELEMENT *
FROM ENV 'ENV1'                TYPE 'P7015'  STA NUM 1
    TO DDNAME SCLSTMTS MEMBER TRANSFER
BUILD
 .
C1BM4100: E010 SUBSYSTEM REQUIRED ON "FROM" CLAUSE FOR THIS ACTION
```

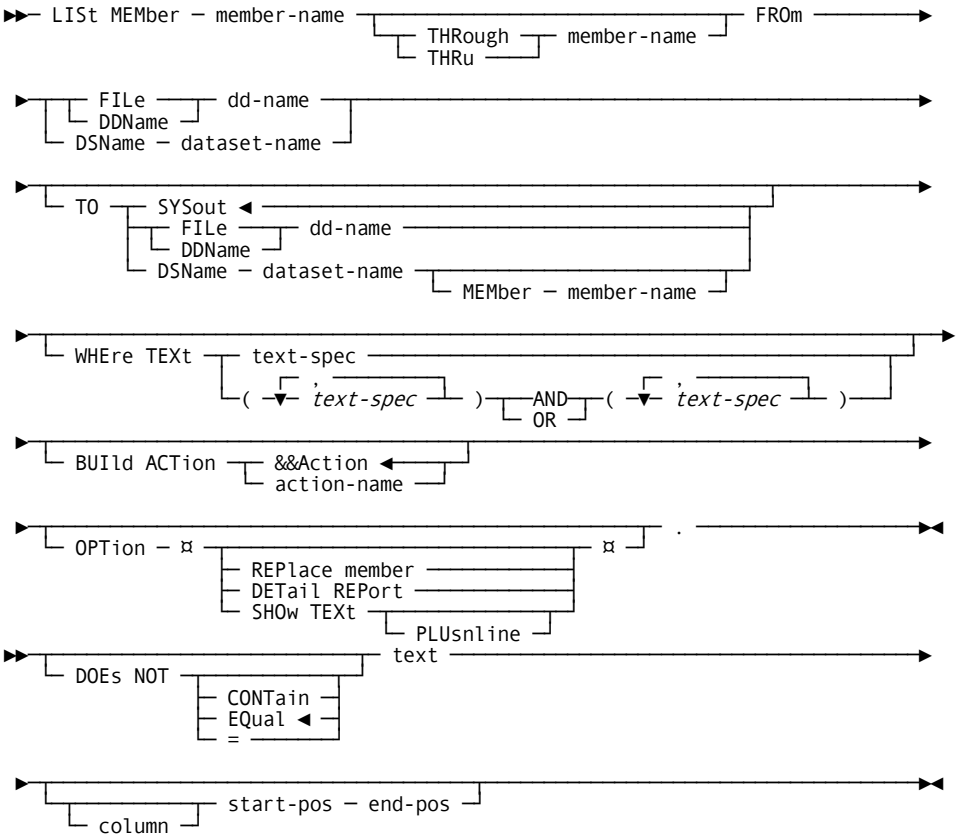
- In the following example the action fails due to the mixed coding (SET and action syntax) of the FROM information. The action is confused and issues message C1G0218E that is not very informative as to the actual issue.

```
SET FROM
    ENV ENV1
    SYS 'P7015'
    SUB 'P7015'
    TYP 'P7015'
    STA NUM 1
.
LIST ELEMENT *
    FROM FILE ARCHIVE
    TO DDNAME SCLSTMTS MEMBER BLDMOV
BUILD ACTION MOVE
 .

C1G0218E  LIST ACTION NOT ALLOWED AGAINST SEQUENTIAL DATA SET
```

List Members from External Library

The LIST MEMBER statement generates a list of elements from an external library.

List Members Syntax



LIST MEMBER member-name

Indicates the member(s) to be listed. Code the required syntax and enter the appropriate member name. In addition, you can use a name-mask with the member name.

THROUGH (THRU) member-name

Indicates that a range of members should be listed, beginning with the member coded in the LIST MEMBER clause, up to and including the member specified in this statement. You can use a name-mask with the member name. If you code the THROUGH clause, you cannot enter a member name in the TO clause.

FROM FILE (DDNAME) dd-name

DSNAME dataset-name

The FROM clause indicates the location of the member being listed. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

If you enter a FILE or DDNAME, be sure the appropriate JCL is coded. The data set you specify in the FROM clause cannot be a load module library or a sequential file.

TO

The TO clause indicates where the element is to be listed. CA Endevor SCM uses both the TO clause in an action and any preceding SET TO clause to determine the "to" criteria for that action.

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.

TO SYSOUT-SYSOUT is the default TO location. If you do not provide a TO clause, and no SET TO information has been coded, CA Endevor SCM writes the action cards to the Execution Report. The Execution Report appears immediately after the LIST action request and cannot be edited.

TO FILE (DDNAME)dd-name or **DSNAME dataset- name**-You can tell CA Endevor SCM to write action cards to both the Execution Report and an external data set by providing a file name (DDname) or a data set name; if you enter a FILE or DDNAME, be sure the appropriate JCL is coded. Use this option if you want to edit the action cards.

TO MEMBER member-name-Enter a member name. This clause is valid only if you are **not** specifying a sequential file.

- CA Endevor SCM ignores a member specification if you have coded the TO SYSOUT option.
- The action fails if you code a member name along with a file (DDname) or data set name that is sequential.

If you are using PDSs and do not provide a member name, CA Endevor SCM assigns a temporary name of TEMPNAME. If you wish to use the temporary naming capability, do not code multiple list requests to the same external data set.

If you are using a PDS and have multiple list statements with only one member name on a SET statement, then all lists go to same member name and only first LIST results are available.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE TEXT text spec-Limits the list to elements that contain (or do not contain) one or more specified 1- to 70-character text strings.

Examples:

```
WHERE TEXT 'WO9-LINKAGE'
```

In this example, CA Endevor SCM lists all elements containing the text string WO9-LINKAGE.

```
Where TEXT  ('COPY COPY005' COLUMN 7 41 AND 'COPY COPY010' COLUMN 7 41)
```

In this example, CA Endevor SCM lists all elements containing the text strings COPY COPY005 and COPY COPY010 between columns 7 and 41 of the element source.

WHERE TEXT DOES NOT CONTAIN 'REMARKS' COLUMN 8 15

In this example, CA Endevor SCM lists all elements that do not contain the text string REMARKS between columns 8 and 15 of the element source.

WHERE TEXT (('M605SUB' OR 'M607SUB') AND DOES NOT CONTAIN 'M606SUB')

In this example, CA Endevor SCM lists all elements that contain either the text string M605SUB or the text string M607SUB and do not contain the text string M606SUB.

BUILD ACTION &&ACTION action-name

Determines the action that appears in the LIST action syntax for the specified element. If you do not enter required BUILD information here, CA Endevor SCM looks for a SET BUILD clause containing the appropriate information. If a SET BUILD clause has not been coded, the system defaults to &&ACTION.

You can enter a specific action (for example, ADD or MOVE) in this clause or the variable &&ACTION. &&ACTION indicates that a specific action will be designated for this element at a later time. This action can be entered manually or using the SET ACTION statement.

OPTIONS

OPTIONS clauses allow you to further specify action requests.

REPLACE MEMBER-When you specify a PDS and member name in the TO clause, list requests fail if the member already exists. Use the REPLACE MEMBER option if you want to replace the existing member in the TO location library.

DETAIL REPORT-By default, in the Execution Report, CA Endevor SCM lists only those elements matching the selection criteria you specify. If you select the DETAIL REPORT option, every element searched is listed in the report-whether or not a match is found.

SHOW TEXT [PLUS n LINES]-This option allows you to print the line of source code that contains a specified text string, plus a designated number of lines of code before and after the text string.

You must code the WHERE TEXT clause if you use the SHOW TEXT option. Otherwise, you receive a syntax error.

Example: List SCL

In the first example, the SCL generates a list of all the elements in the Payroll Reporting subsystem that contain the text "COPY PAYCOPY1" in columns 7 through 45, inclusive. SCL will be generated for each element found.

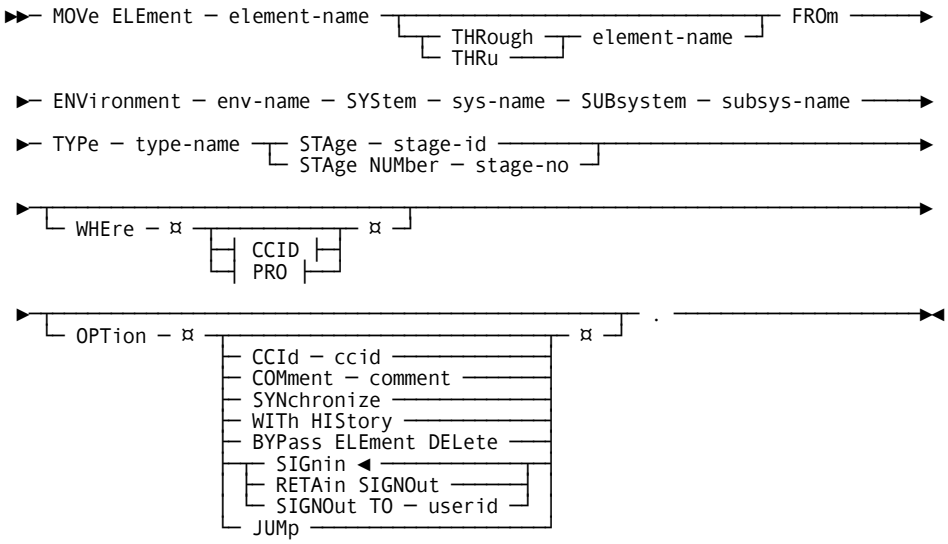
```
LIST ELEMENT '*'

    FROM ENVIRONMENT 'PROD'
                SYSTEM 'PAYROLL'
                SUBSYSTEM 'REPORTS'
                TYPE 'COBOL'
                STAGE NUMBER 1
    WHERE TEXT EQ 'COPY PAYCOPY1' COLUMN 7 45
    BUILD ACTION GENERATE .
```

In the second example, the SCL generates a list of all members in PAYROLL.SRCLIB that begin with "PAYRPT*' and contain the string "COPY PAYCOPY3" in columns 7 through 45, inclusive. The subsequent report will display the entire line in which the text was found.

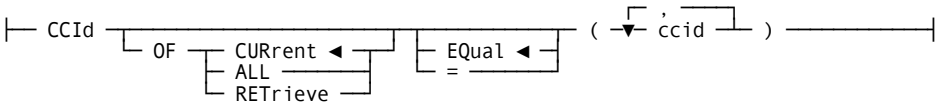
```
LIST MEMBER 'PAYRPT*'
       FROM DSNAME 'PAYROLL.SRCLIB'
       WHERE TEXT EQ 'COPY PAYCOPY3' IN COLUMN 7 45
       OPTIONS SHOW TEXT .
```

Move Statement

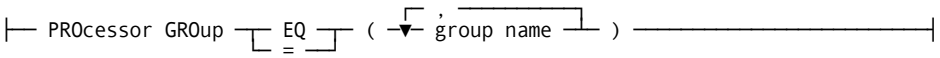
The MOVE statement moves elements between inventory locations along a map.



Expansion of CCID



Expansion of PRO



MOVE ELEMENT element-name

Indicates the element(s) to be moved. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name.

THROUGH (THRU) element-name

Indicates that a range of elements should be moved, beginning with the element coded in the MOVE ELEMENT statement, up to and including the element specified in this statement. You can use a name-mask with the element name.

FROM ENVIRONMENT env-name

SYSTEM sys-name

SUBSYSTEM subsys-name

TYPE type-name

STAGE stage id

STAGE NUMBER stage-no

The FROM clause indicates the location of the element being moved. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

You must specify an environment, system, subsystem, type, and stage. You can use a name-mask with the system, subsystem and type. The environment name and stage information must be explicit. The stage specification can be either one of the following:

- STAGE ID-Enter a single alphanumeric stage identifier.
- STAGE NUMBER-Enter either **1** or **2**.

Moving sourceless elements: When the Move element is sourceless, the element's last level timestamp is checked against the next sourced element found up the map. If the timestamps are the same, a sourceless element record will be created at the target location if no element exists. The processor uses the element source data from the next sourced element for its GEN process. If the timestamps are not the same, the move action fails. You will need to take corrective action at this point. You can either delete the sourceless element or regenerate the sourceless element and then follow this by a move.

When the target location has a sourceless element, its last level timestamp is compared with the next sourced element. If the timestamps are not equal, the action fails unless the SYNC option is used.

WHERE

Use the WHERE clause to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID OF ccid -Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

- **CURRENT-**Tells CA Endevor SCM to look through the CCID fields in the MCF (Master Control File) to find a specified CCID(s). This is the default.
- **ALL-**Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s). If the element is sourceless , the source element delta level CCID checks are not done, because the element source and delta files are not present.
- **RETRIEVE-**Tells CA Endevor SCM to use the CCID in the Master Control File RETRIEVE CCID field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1:  WHERE CCID OF CURRENT (PROJ001, PROJ002, PROJ004)
Example 2:  WHERE CCID OF ALL (PROJ00V)
```

WHERE PROCESSOR GROUP group name**-**This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1:  WHERE PROCESSOR GROUP  (COBVS, COBII)
Example 2:  WHERE PROCESSOR GROUP  (COBV)
```

OPTIONS

OPTIONS clauses allow you to further specify requests.

CCID ccid/COMMENT comment-You can enter a 1- to 12- character CCID and/or a 1- to 40-character comment.

CCIDs and/or comments may be required. If you do not provide a required CCID and/or comment, the MOVE action fails.

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

When you specify a CCID and/or comment in a MOVE action without history, CA Endevor SCM uses this CCID and/or comment to set the last action CCID and/or COMMENT fields. CA Endevor SCM also:

- Sets the source and generate CCID and/or COMMENT fields to their value at the source location of the move.
- Sets the source and component list delta CCID and/or COMMENT fields to their last value at the source location of the move.
- Clears the retrieve CCID and/or COMMENT field.

When you specify a CCID and/or comment in a MOVE action using the WITH HISTORY option, CA Endevor SCM uses this CCID and/or comment to set the last action CCID and/or COMMENT fields.

CA Endevor SCM also does the following:

- Sets the source and generate CCID and/or COMMENT fields to their value at the source location of the move.
- Clears the retrieve CCID and/or COMMENT fields.
- Moves source delta CCIDs and comments with their respective delta levels.
- Moves the current version/level of the Component List.

  For more information, see Monitoring Components in a Move Processor in the *Automated Configuration Option Guide*.

SYNCHRONIZE- The SYNCHRONIZE option compensates for differences between the base level of a source element and the current level of a target element. CA Endevor SCM attempts to find a sync level between the source and target elements, beginning with the first level at the source, and working forward through the deltas. If CA Endevor SCM finds a sync level, it compares the two and creates a new level at the target that reflects the differences. If CA Endevor SCM cannot find a sync level and you specify SYNC, CA Endevor SCM issues an out of sync message. CA Endevor SCM then compares the last level of the source and last level of the target, and creates a new level at the target that reflects the differences.

When moving with history, if the sync point is found, CA Endevor SCM moves the element from the FROM location to the TO location, appending the FROM location delta levels after the sync-point element. If the two levels are different, and SYNC is specified, CA Endevor SCM first creates a sync level at the target reflecting the differences between the base level of the FROM element and the target, then moves the element to the TO location and appends the FROM location delta levels to the target.

WITH HISTORY-The WITH HISTORY option preserves source element change history. If you request MOVE WITH HISTORY, CA Endevor SCM first ensures that the current level of the target element is the same as the base level of the source element. It then moves all levels of the element from source to target, appending the source change history to the target change history.

If you do not code this option, CA Endevor SCM moves the element(s) without history. When you move the element without history CA Endevor SCM searches through the element levels at the source location to find a matching level at the target location. CA Endevor SCM then compares the two and creates a new level at the target location that reflects the differences.

If the base level of the source element differs from the current level at the target, the move fails unless you code the SYNCHRONIZE option.

BYPASS ELEMENT DELETE-This option tells CA Endevor SCM to retain the element in the source stage after successfully completing the move.

SIGNIN-Default. This option tells CA Endevor SCM to sign in all elements at the target stage after successfully completing the move. You must code this option to override SET OPTION RETAIN SIGNOUT or SET OPTION SIGNOUT TO clauses.

RETAIN SIGNOUT-This option tells CA Endevor SCM to retain the source location signouts for all elements at the target location. This option applies only if the element was signed out at the target before the MOVE.

If the element was signed out at the target before the MOVE, it will be signed out to that same ID-at the target-after the MOVE.

If the element was not signed out at the target before the MOVE, it will not be signed out at the target after the MOVE.

If you do not use this option, the element at the target location is not signed out, regardless of whether it was signed out at the target before the MOVE took place.

SIGNOUT TO userid-This option tells CA Endevor SCM to sign all elements out to the specified user ID at the target stage.

JUMP-The JUMP option tells CA Endevor SCM to move elements across environments even if the element exists at an intermediate stage that is not on the map. If the element exists at an intermediate stage, the move fails if REQ ELM JUMP ACKNOWLEDGE=Y at the system level and the JUMP option is not coded.

In either case, CA Endevor SCM issues a message informing you that the element exists in a non-map stage between the source and target stages of the move.

Example: Move SCL

This SCL moves an element from Stage 1. The element history will be retained at the target stage.

```
MOVE ELEMENT 'PAYRPT17'
     FROM ENVIRONMENT 'PROD'
          SYSTEM 'PAYROLL'
          SUBSYSTEM 'REPORTS'
          TYPE 'COBOL'
          STAGE NUMBER 1
     OPTIONS CCID REQ#43034
          COMMENT 'MOVE INTO PRODUCTION'

          WITH HISTORY.
```

Print Statement

The PRINT statement prints selected information about elements or library members, depending on the data entered in the FROM clause.

Printing from CA Endevor SCM

When executing the PRINT action against CA Endevor SCM, you can request the following information about elements and component lists:

- BROWSE (the default) prints all statements in the specified level of the element, as well as the level at which each statement was inserted.
- CHANGES shows all inserts and deletes made to the element at the level specified.
- HISTORY prints all statements in all levels of the element.
- SUMMARY prints one line of summary information for each level.

You can request the following information about elements only:

- MASTER prints Master Control File information for the element.
- LISTING prints the output listing for the element. ACM is required.

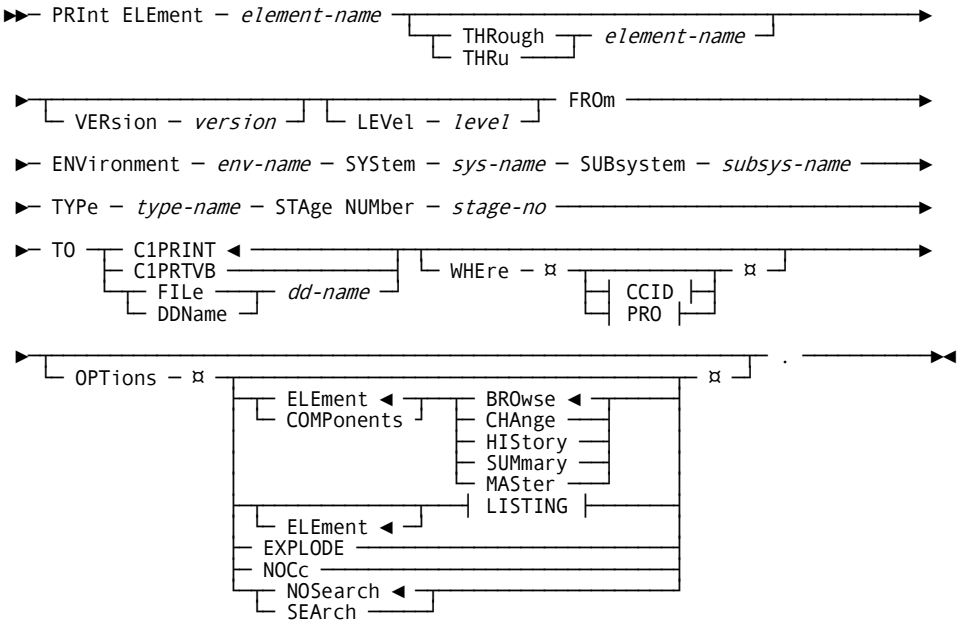
Printing from an Output Library

When you execute the PRINT action against an output library, the source of the selected, footprinted member is printed.

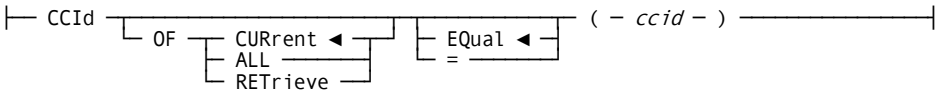
The Print Element Statement

The PRINT ELEMENT statement prints selected information about the element you specify.

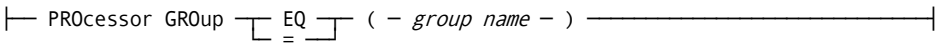
Print Element Syntax



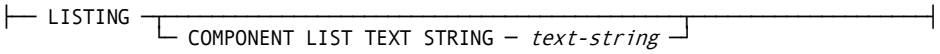
Expansion of CCID



Expansion of PRO



Expansion of LISTING



PRINT ELEMENT element-name

Indicates the up to ten character name of the elements to be printed. You can use a name-mask, unless you specify a level (in the LEVEL clause).

THROUGH (THRU) element-name

Indicates that a range of elements should be printed, beginning with the element coded in the PRINT ELEMENT statement, up to and including the element specified in this statement. You can use a name-mask with either name. If you enter a THROUGH clause, you cannot enter a LEVEL clause.

VERSION version LEVEL level

Indicates the version and level number you want to print. You must code a full element name if you want to indicate a version level. Acceptable version numbers are **1-99**. Acceptable level numbers are **00-99**. By default, information for the current version level is printed. VERSION and LEVEL must be specified together. If you enter this clause, you cannot use the THROUGH clause.

FROM **ENVIRONMENT *env-name***

SYSTEM *sys-name*

SUBSYSTEM *subsys-name*

TYPE *type-name*

STAGE *stage-no*

The FROM clause indicates the location of the element being printed. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the “from” criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

Note: For more information about the SET FROM statement, see the chapter "Using Set, Clear, and EOF Statements (see page 49)."

You must specify an environment, system, subsystem, type, and stage number (either **1** or **2**). The environment name must be explicit. You can use a name-mask with the system, subsystem, type, and stage number.

TO **C1PRINT | C1PRTVB | FILE *dd-name* | DDNAME *dd-name***

The TO clause indicates where the element prints.

- A TO clause in an action overrides values in a SET TO clause that precedes the action. For more information about the SET TO statement, see Set To Syntax (see page 70)."
- If you do not enter any information in the TO clause for the PRINT action, any preceding SET TO statement information is used.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.
- If you do not enter any information in the TO clause for the PRINT action, and if a SET TO clause has not been entered, the system defaults to C1PRINT.

C1PRINT

Prints to an *internal* print queue or file. C1PRINT requires the appropriate JCL. If you print an element with records that are longer than 121 characters, then the use of C1PRINT can lead to truncation of the output. To avoid this, use the C1PRTVB ddname. Examples follow:

- To send your output to the queue, code the following. This is the default:

```
//C1PRINT DD SYSOUT=*
```

- To send your output to an existing file, code the following:

```
//C1PRINT DD DISP=SHR,DSN=filename
```

C1PRTVB

Prints elements that have records that are longer than 121 characters to an *internal* print queue or file. You must have previously allocated the C1PRTVB data set appropriately. The recommended DCB for C1PRTVB is LRECL=27994,BLKSIZE=0,RECFM=VB. However, if the record size of any record is longer than 27978, code a larger record length. You must code a sufficiently long LRECL for the output file. The LRECL size should be at least 16 bytes longer than the longest record of the element. For example, allocate the data set as follows:

```
//C1PRTVB  DD DISP=(,CATLG),DSN=my.C1PRTVB,
//         SPACE=TRK,(5,5),UNIT=SYSDA,
//         DCB=(RECFM=VB,LRECL=27994,BLKSIZE=0)
```

FILE *dd-name* | DDNAME *dd-name*

Prints to a location *external* to CA Endevor SCM (for example, a library, sequential file, or PDS). When you enter a file name or DDname, be sure that the appropriate JCL is coded. For the ddname value, you can specify a sequential file with a mandatory fixed record format and an LRECL of 133. If these specifications are not defined, the PRINT action fails. If you print an element with records that are longer than 121 characters, then the use of this data set can lead to truncation of the output. To avoid this, use the C1PRTVB ddname.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the “where” criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- IF the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

Note: For more information about the SET WHERE statement, see the chapter "Using Set, Clear, and EOF Statements (see page 49)."

WHERE CCID OF ccid

Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

- **CURRENT-**Tells CA Endevor SCM to look through the CCID fields in the MCF (Master Control File) to find a specified CCID(s). This is the default.
- **ALL-**Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s). If the element is sourceless , the source element delta level CCID checks are not done, because the element source and delta files are not present.
- **RETRIEVE-**Tells CA Endevor SCM to use the CCID in the Master Control File's RETRIEVE CCID field. If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The following example illustrate the use of this clause:

```
Example  1: WHERE CCID OF CURRENT (PROJ001, PROJ002, PROJ004)
Example 2: WHERE CCID OF ALL (PROJ00V)
```

WHERE PROCESSOR GROUP group name

This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if and separating them with commas. The processor groups may extend over multiple lines if necessary.

The next examples illustrate the use of this clause:

```
Example 1: WHERE PROCESSOR GROUP (COBVS, COBII)
Example 2: WHERE PROCESSOR GROUP (COBV
```

OPTIONS

OPTIONS clauses allow you to further specify an action request.

NOCC

Suppresses the default printing of a header on each page of output. For the PRINT ELEMENT MASTER output, other line feed characters in column 1 of the output are replaced with a blank space.

ELEMENT | COMPONENTS BROWSE | CHANGES | HISTORY | SUMMARY | MASTER

Prints element or component information.

ELEMENT

Prints all information for the specified element, but not its related components. The default.

COMPONENTS

Prints all component information for the specified element. The *CA Endevor SCM Automated Configuration* option is required to use the COMPONENTS keyword.

CA Endevor SCM prints as much information as is available for the component list. For example, if you code COMPONENTS CHANGES but there were no changes to the output components section, that section would not appear in the associated listing.

You can specify one, but only one, of the following options in the clause. BROWSE is the default.

- **BROWSE—** Prints the current element or component list source, indicating the level at which each line was added. If you specify a particular level, CA Endevor SCM prints the source for that level. The Default.
- **CHANGES—** Prints all the changes-inserts and deletes-made to the element or component list at the level specified. If you do not specify a level in the LEVEL clause, changes for the current level of the element are shown.
- **HISTORY—** Prints all lines that have ever been in the element or component list source, noting the level at which the line was added, changed, or deleted. If you specify a level in the LEVEL clause, CA Endevor SCM prints history for that level.
- **SUMMARY—** Prints a summary line of data for each level of the element or component list specified, and includes information appropriate to that level (for example, the number of inserts and the number of deletes).
- **MASTER—** Prints Master Control File information stored for the selected element, as well as the BROWSE data, which is the current data pertaining to that element or component (such as last processor, processor return codes, current version or level, and so on.) If you code COMPONENTS with the MASTER option, CA Endevor SCM prints the element MASTER information and the component BROWSE data.

The BROWSE, CHANGES, HISTORY, SUMMARY, and MASTER printouts provide the same information as their corresponding online panels. For information about the online panels, see the *User's Guide*.

The element BROWSE, CHANGE, HISTORY and SUMMARY options are not supported on sourceless elements. When encountered, the action is skipped and an information message is written to the action log. The MASTER option is supported for both sourced and sourceless elements.

ELEMENT LISTING COMPONENT LIST TEXT STRING text-string

Prints the element's associated output listing. The *CA Endevor SCM Automated Configuration Option* is required. Specifying the keyword ELEMENT is optional. The default value for the LISTING option is LIST. If you want to use the default value, then do **not** specify COMPONENT LIST TEXT STRING text-string. You can override the default by specifying a text string using the keyword COMPONENT LIST TEXT STRING, or by changing the C1DEFLTS parameter COMPLISTWD=, which defines the site-wide component listing string ID.

COMPONENT LIST TEXT STRING *text-string*

Specifies the one- to eight-character text-string used to identify the listing data set to print.

To identify the output listing file to be printed, the LISTING option's value is matched against the contents of all the component output data sets. To find the output listing library, all component output data sets are searched for a particular string. The first one found containing this string becomes the targeted listing data set. All component output data sets are searched for a match in the following order:

- The *text-string* you specify with the COMPONENT LIST TEXT STRING keyword. If you did not use this option, or the string is not found, then
- The COMPLISTWD= parameter in the C1DEFLTS table. The default value for this parameter is LIST. The default value can be changed on the C1DEFLTS table. You can also change the text string by changing the value for the LISTING DATASET STRING on the User Default panel.

If you code the LISTING option, you should not code the following options:

NOCC— Coding NOCC results in a syntax error.

COMPONENTS

BROWSE

CHANGE

HISTORY

SUMMARY

MASTER

If you code multiple display options (browse, change, history, summary), then the last one coded is returned.

SEARCH or NOSEARCH

The SEARCH option tells CA Endevor SCM to look and print all occurrences of the element on the map.

The default is NOSEARCH. Code NOSEARCH to restrict CA Endevor SCM's search to the current environment.

EXPLODE

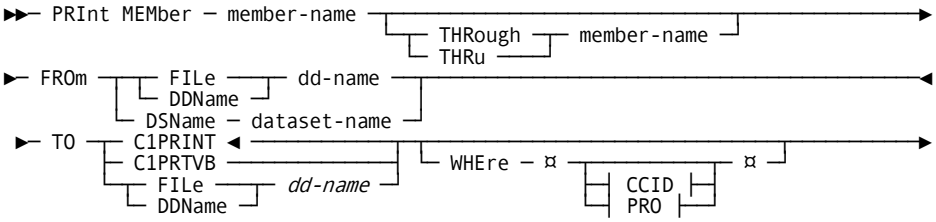
Prints the input component information extracted from the ACMQ files (ACMROOT and ACMXREF), for the specified element.

When Explode is specified, the print action for the element jumps to the top of the list to be executed. Therefore, any Print actions that use the Explode option are processed first before other actions are processed in Type sequence order.

The Print Member Statement

The PRINT MEMBER statement prints selected information about the member you specify. You can print from either CA Endevor SCM or from selected output libraries.

Print Member Syntax



PRINT MEMBER member-name

Indicates the name of the member(s) to be printed. You can use a name-mask.

THROUGH (THRU) member-name

Indicates that a range of members should be printed beginning with the member coded in the PRINT MEMBER statement, up to and including the member specified in this statement. You can use a name-mask with either name.

FROM FILE (DDNAME) dd-name

DSNAME dataset-name

The FROM clause indicates the location of the member being printed. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

You must enter a FILE, DDNAME, or DSNAME (enter one and only one); be sure the appropriate JCL is coded for a FILE or DDNAME. If you enter any other information in the FROM clause, it is ignored.

TO C1PRINT | C1PRTVB | FILE *dd-name* | DDNAME *dd-name*

The TO clause indicates where the element prints.

- A TO clause in an action overrides values in a SET TO clause that precedes the action. For more information about the SET TO statement, see Set To Syntax (see page 70)."
- If you do not enter any information in the TO clause for the PRINT action, any preceding SET TO statement information is used.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.
- If you do not enter any information in the TO clause for the PRINT action, and if a SET TO clause has not been entered, the system defaults to C1PRINT.

C1PRINT

Prints to an *internal* print queue or file. C1PRINT requires the appropriate JCL. If you print an element with records that are longer than 121 characters, then the use of C1PRINT can lead to truncation of the output. To avoid this, use the C1PRTVB ddname. Examples follow:

- To send your output to the queue, code the following. This is the default:

```
//C1PRINT DD SYSOUT=*
```

- To send your output to an existing file, code the following:

```
//C1PRINT DD DISP=SHR,DSN=filename
```

C1PRTVB

Prints elements that have records that are longer than 121 characters to an *internal* print queue or file. You must have previously allocated the C1PRTVB data set appropriately. The recommended DCB for C1PRTVB is LRECL=27994,BLKSIZE=0,RECFM=VB. However, if the record size of any record is longer than 27978, code a larger record length. You must code a sufficiently long LRECL for the output file. The LRECL size should be at least 16 bytes longer than the longest record of the element. For example, allocate the data set as follows:

```
//C1PRTVB  DD DISP=(,CATLG),DSN=my.C1PRTVB,
//         SPACE=TRK,(5,5),UNIT=SYSDA,
//         DCB=(RECFM=VB,LRECL=27994,BLKSIZE=0)
```

FILE *dd-name* | DDNAME *dd-name*

Prints to a location *external* to CA Endevor SCM (for example, a library, sequential file, or PDS). When you enter a file name or DDname, be sure that the appropriate JCL is coded. For the ddname value, you can specify a sequential file with a mandatory fixed record format and an LRECL of 133. If these specifications are not defined, the PRINT action fails. If you print an element with records that are longer than 121 characters, then the use of this data set can lead to truncation of the output. To avoid this, use the C1PRTVB ddname.

Example: Print SCL

In the first example, the SCL prints the current version of element "PAYRPT19." The output is written to the default DDname (C1PRINT).

```
PRINT ELEMENT 'PAYRPT19'

    FROM ENVIRONMENT 'PROD'
                 SYSTEM 'PAYROLL'
                 SUBSYSTEM 'REPORTS'
                 TYPE 'COBOL'
                 STAGE NUMBER 1 .
```

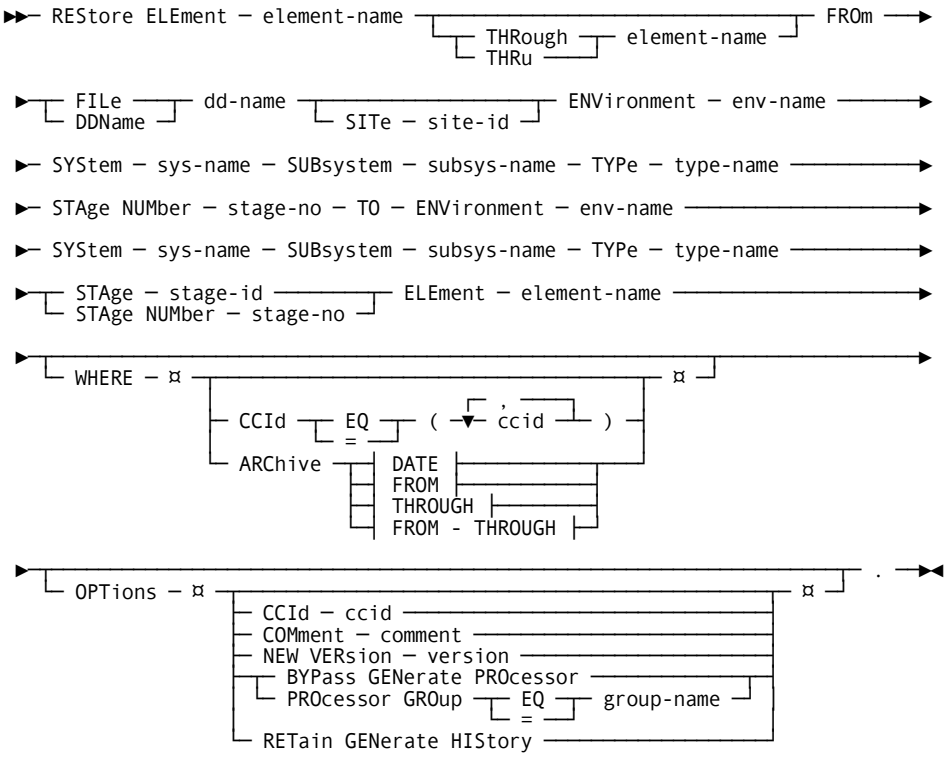
The SCL in the second example prints member "PAYRPT12" from the CA Endevor SCM Listing Library. The output is sent to the default Ddname (C1PRINT).

```
PRINT MEMBER 'PAYRPT12'
        FROM DSNAME 'ENDEVOR.PAYROLL.STAGE1.LISTINGS' .
```

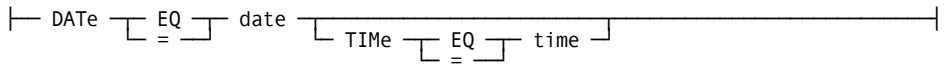
Restore Statement

The Restore statement restores an element from an archive data set back to CA Endevor SCM, *copying* the source as it was before the element was archived or transferred to the data set. The RESTORE action can restore elements from an unload CA Endevor SCM file.

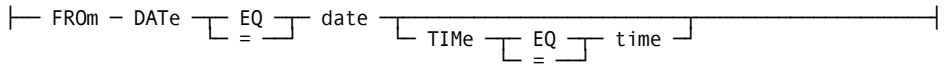
The Restore action is available in batch only.



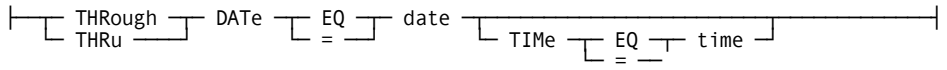
Expansion of DATE



Expansion of FROM



Expansion of THROUGH



RESTORE ELEMENT *element-name*

Indicates the element(s) to be restored. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name.

THROUGH (THRU) *element-name*

Indicates that a range of elements should be restored, beginning with the element coded in the RESTORE ELEMENT statement, up to and including the element specified in this statement. You can use a name-mask with the element name. If you use the THROUGH clause, however, you cannot enter a new element name (in the TO clause).

FROM FILE (DDNAME) *dd-name* **SITE** *site-id*
ENVIRONMENT *env-name*
SYSTEM *system-name*
SUBSYSTEM *subsys-name*
TYPE *type-name*
STAGE NUMBER *stage-no*

The FROM clause indicates the location of the element being restored. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

Note: For a description of the SET FROM statement, see the chapter "Using Set Clear, and EOF Statements."

The first clause of the FROM statement must be a FILE or DDNAME, indicating from which file the element is being restored. It must be followed by environment, system, subsystem, type, and stage number (either 1 or 2). The environment name must be explicit. You can use a name-mask with the system, subsystem, type, and stage number.

The FROM file can be either an archive or unload file, or a concatenation of archive files or unload files. You cannot concatenate archive and unload files in a single RESTORE action.

Note: If you code more than one RESTORE statement in your SCL, each RESTORE statement can point to a different DDNAME and a different file type (archive or unload).

Entering a site ID is optional. This field further defines the location of the element being restored.

TO **ENVIRONMENT** *env-name*
SYSTEM *system-name*
SUBSYSTEM *subsys-name*
TYPE *type-name*
STAGE *stage-id*
STAGE NUMBER *stage-no*
ELEMENT *element-name*

The TO clause indicates where the element is being restored. CA Endevor SCM uses both the TO clause in an action and any preceding SET TO clause to determine the "to" criteria for that action.

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.

If no SET TO clause has been coded, CA Endevor SCM retrieves the required information from the FROM clause coded for this action. Environment must be coded first in TO.

You must specify an environment, system, subsystem, type, and stage. The stage specification can be either one of the following:

- STAGE ID-Enter a single alphanumeric stage identifier.
- STAGE NUMBER-Enter either **1** or **2**.

Remember that you cannot use a name-mask with a TO field name. Enter a different element name if you want to change the element name specified (that is, the archived element name) in the RESTORE ELEMENT clause. If you do not enter an element name here, CA Endevor SCM uses the archived element name.

- You can enter a new element name only if a full element name was coded in the RESTORE ELEMENT clause; that is, if you have not used a name-mask.
- If you enter an element name here, you cannot use the THROUGH clause.
- If you want to code a different element name, you must do so in the RESTORE statement. The SET TO MEMBER clause does not apply to this action.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID *ccid* **-**Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The following examples illustrate the use of this clause:

```
Example 1: WHERE CCID EQ PROJ__V
Example 2: WHERE CCID (PROJ__1, PROJ__2, PROJ__4)
```

WHERE ARCHIVE-This clause allows you to select elements based on the date and, optionally, time that an element was archived. There are four possible forms for this clause:

WHERE ARCHIVE DATE *mm/dd/yy* **[TIME** *hh:mm* **]**

This clause tells CA Endevor SCM to archive only those elements with this date, and optionally, time stamp.

WHERE ARCHIVE FROM DATE *mm/dd/yy* **[TIME** *hh:mm***]**

This clause tells CA Endevor SCM to archive all elements with a date and, optionally, time stamp on or after the specified date and time stamps.

WHERE ARCHIVE THROUGH DATE *mm/dd/yy* **[TIME** *hh:mm* **]**

This clause tells CA Endevor SCM to archive all elements with a date and, optionally, time stamp earlier than and including the specified date and time stamp.

WHERE ARCHIVE FROM DATE *mm/dd/yy* **[TIME** *hh:mm* **]**

THROUGH DATE mm/dd/yy [TIME *hh:mm* **]**

This clause tells CA Endevor SCM to archive only those elements with date, and optionally, time stamps within the specified range.

Note: If you enter a time, you must enter a date with it.

OPTIONS

CCID *ccid***/COMMENT** *comment*-You can enter a 1- to 12-character CCID and/or a 1- to 40-character comment.

CCIDs and/or comments may be required. If you do not provide a required CCID and/or comment, the RESTORE action fails.

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

- Set the generate and component list delta CCID and/or COMMENT fields if the generate processor is run. CA Endevor SCM writes this comment to the Master Control File replacing the comment for that element. CA Endevor SCM does not set these fields if you code BYPASS GENERATE PROCESSOR.
- Set the last action CCID and/or COMMENT fields. CA Endevor SCM sets the source, source delta, and RETRIEVE CCID and/or COMMENT fields based on the archive data set.

CA Endevor SCM sets the source, source delta, and RETRIEVE CCID and/or COMMENT fields based on the archive data set.

NEW VERSION *version*-Tells CA Endevor SCM to assign the specific version number to the element. Acceptable values are **1-99**. If the element exists at the target location or at a location up the mapped route, the RESTORE action fails.

BYPASS GENERATE PROCESSOR-Tells CA Endevor SCM not to execute the generate processor after restoring the element.

PROCESSOR GROUP EQ/= *group name***-**Tells CA Endevor SCM which processor group to associate with the restored element. The keywords PROCESSOR GROUP EQ and BYPASS GENERATE PROCESSOR are mutually exclusive (you can code one or the other). If both are coded, the default BYPASS GENERATE PROCESSOR is used.

RETAIN GENERATE HISTORY – Retain the Master Control File last generate information. This option when used with the Transfer to Archive action can be useful when converting elements to a new delta type.

Example: Restore SCL

This SCL restores all of the COBOL elements from the archive file associated with the ARCHIN DD statement that you specify in the execution JCL.

```
RESTORE ELEMENT ' '

  FROM FILE ARCHIN
               ENVIRONMENT 'PROD'
               SYSTEM 'PAYROLL'
               SUBSYSTEM 'REPORTS'
               TYPE 'COBOL'
               STAGE NUMBER 1
  TO           ENVIRONMENT 'PROD'
               SYSTEM 'PAYROLL'
               SUBSYSTEM 'REPORTS'
               TYPE 'COBOL'
               STAGE NUMBER 1
  OPTIONS      CCID REQ344145

               COMMENT 'ARCHIVE REPORTING SUBSYSTEM PROGRAMS'
```

How Restore Processing Works

The processing sequence for the Restore action follows:

1. CA Endevor SCM determines whether the element is present in the stage indicated. If the element exists in that stage, the Restore action fails.
2. Restores the element, if it is not present, to the specified location (environment, stage, system, subsystem, and type). This includes:
   - Restoring all base and delta levels for the element.
   - Copying the element definition to the Master Control File.
   - Restoring component lists, if present.

   If you use the NEW VERSION option, CA Endevor SCM assigns the new version number to the restored element. Otherwise, the element retains its version number from the file.
3. Updates the Master Control File.
4. Continues based on the value in the BYPASS GENERATE PROCESSOR field. If the value in this field is Y, CA Endevor SCM does not generate the element.

   If the value in this field is N, CA Endevor SCM:
   - Determines the processor group to use. It then executes the generate processor in that group if one has been specified.

     Note: If you are performing multiple element restores, all of the elements will be restored before the generate processors are executed.
   - Reads the type definition for a source output library specification, then writes a copy of the current level of the element to that library. If EXPAND INCLUDES = Y, Endevor expands INCLUDE statements in the source. After the generate processor has been run for the element, Endevor updates the information in the Master Control File.
5. Updates the Master Control File after the RESTORE action is successfully completed.

How to Allocate Data Sets for Large Elements

When you use the C1BM3000 utility to Transfer or Restore an element, the data sets are dynamically allocated to process data. The dynamically allocated data sets may not be sufficient, if a large element is transferred, and the System abort B37 error may occur. In this case, you can use the C1WORK01—C1WORK06 DD statements to override the standard dynamic allocation of data sets, and run the Transfer or Restore action again. These DD statements override the following records during the Transfer or Restore actions:

- C1WORK01—Merged configuration records
- C1WORK02—Configuration delta records
- C1WORK03—Configuration base records
- C1WORK04—Merged element records
- C1WORK05—Element delta records
- C1WORK06—Element base records

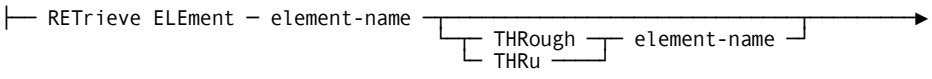
You can code any or all of the previous DD statements. If you code one C1WORK DD statement, then it overrides the equivalent dynamic allocation of datasets. For example, if a very large element is transferred the regular dynamic allocation of datasets may not be sufficient for storing element base, element delta, and merger element records. In this case, you can code the C1WORK04, C1WORK05 and C1WORK06 statements to override the dynamic allocation of datasets.

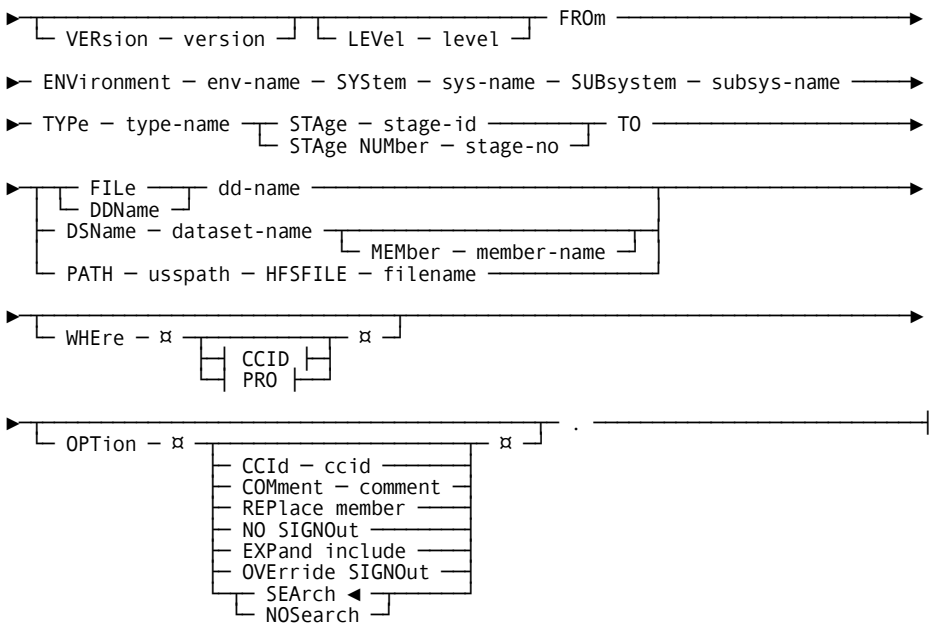
A sample JCL illustrating how to code these DD statements is shown as follows:

```
//STEP1     EXEC PGM=C1BM3000,PARM='SCLIN,MSGOUT1,,MSGOUT2'
//MSGOUT1   DD  SYSOUT=A
//MSGOUT2   DD  SYSOUT=A
//SCLIN     DD  *
//ARC       DD  DISP=OLD,DSN=ARCHIVE.DATASET
//C1WORK04  DD  UNIT=SYSDA,SPACE=(CYL,(50,50)),
//              DCB=(RECFM=VB,LRECL=5000,BLKSIZE=6233)
//C1WORK05  DD  UNIT=SYSDA,SPACE=(CYL,(50,50)),
//              DCB=(RECFM=VB,LRECL=5000,BLKSIZE=6233)
//C1WORK06  DD  UNIT=SYSDA,SPACE=(CYL,(50,50)),
//              DCB=(RECFM=VB,LRECL=5000,BLKSIZE=6233)
//BSTIPT01  DD  *
```

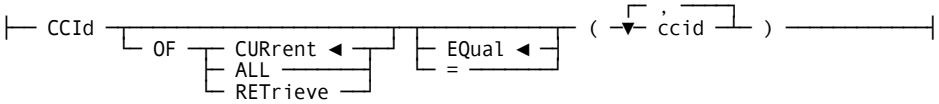
Retrieve Statement

The RETRIEVE statement copies an element to a user data set.

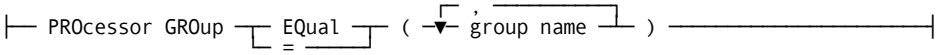




Expansion of CCID



Expansion of PRO



RETRIEVE ELEMENT element-name

Indicates the name of the element(s) to be retrieved. You can specify the element name using a name-mask, unless you want to retrieve a specific level of the element.

THROUGH (THRU) element-name

Indicates that a range of elements should be retrieved, beginning with the element coded in the RETRIEVE ELEMENT statement, up to and including the element specified in this statement. You can use a name-mask with the element name.

If you use the THROUGH clause, you cannot enter a member name in the TO clause or a different level in the LEVEL clause.

VERSION version LEVEL level

Indicates the version and level number you want to retrieve. You must code a full element name if you want to indicate a version level. You cannot enter this clause if you use the THROUGH clause. Acceptable version numbers are **1-99**. Acceptable level numbers are **00-99**. By default, information for the current version level is retrieved. VERSION and LEVEL must be specified together. If you enter this clause, you cannot use the THROUGH clause.

FROM **ENVIRONMENT env-name**
SYSTEM system-name
SUBSYSTEM subsys-name
TYPE type-name
STAGE stage-id
STAGE NUMBER stage-no

The FROM clause indicates the location of the element being retrieved. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

You must specify an environment, system, subsystem, type, and stage. The environment name must be explicit. You can use a name-mask with the system, subsystem, type and stage. The stage specification can be either one of the following:

- STAGE ID-Enter a single alphanumeric stage identifier.
- STAGE NUMBER-Enter either **1** or **2**.

If you use a name-mask with the stage, CA Endevor SCM begins searching for the specified element in Stage 1 of the current environment, and retrieves the first element that matches the specified element name, regardless of its location, version or level.

If the element is sourceless, the source is retrieved from the next sourced element found up the map. You will get a warning message if the last level timestamp on the retrieved element is not the same as the sourceless element.

TO **FILE (DDNAME) dd-name**
DSNAME dataset-name
MEMBER member-name
PATH usspath
HFSFILE filename

The TO clause indicates where the element is being retrieved. CA Endevor SCM uses both the TO clause in an action and any preceding SET TO clause to determine the "to" criteria for that action.

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.

You must enter a FILE, DDNAME, DSNAME or PATH in conjunction with HFSFILE (enter one and only one). If you enter either a FILE or DDNAME, be sure the appropriate JCL is coded.

Enter a member name if it differs from the element name specified in the RETRIEVE ELEMENT clause. Remember that you cannot use a name-mask with a TO field name.

If you do not enter a member name, CA Endevor SCM assumes that the element name and member name are the same. If you code a member name:

- The RETRIEVE ELEMENT clause must contain a fully qualified element name.
- You cannot use the THROUGH clause.

The SET TO MEMBER clause does not apply to the RETRIEVE action.

PATH

The USS directory to which you want to retrieve the element. This has a maximum of 768 characters.

HFSFILE

The name of the file for the retrieved element. The file name has a maximum of 255 characters.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID OF ccid-Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

- **CURRENT**-Tells CA Endevor SCM to look through the CCID fields in the Master Control File to find a specified CCID(s). This is the default.
- **ALL-**Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s). If the element is sourceless, the source element delta level CCID checks are not done.
- **RETRIEVE-**Tells CA Endevor SCM to use the CCID in the Master Control File RETRIEVE CCID field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The following examples illustrate the use of this clause:

```
Example 1: WHERE CCID OF CURRENT (PROJ__1, PROJ__2, PROJ__4)
Example 2: WHERE CCID OF ALL (PROJ__V)
```

WHERE PROCESSOR GROUP *group name***-** This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The following examples illustrate the use of this clause.

```
Example 1: WHERE PROCESSOR GROUP (COBVS, COBII)
Example 2: WHERE PROCESSOR GROUP (COBV)
```

OPTIONS

OPTIONS clauses allow you to further specify action requests.

CCID ccid/COMMENT comment-You can enter a 1- to 12-character CCID and/or a 1- to 40-character comment. CCIDs and/or comments may be required. If you do not provide a required CCID and/or comment, the RETRIEVE action fails.

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

REPLACE MEMBER-If you retrieve an element to a library, CA Endevor SCM checks to see whether that element is currently in the library. By default, if this condition exists, the request will be rejected. The REPLACE MEMBER option, however, enables you to replace the member currently in the library with the retrieved element. Specify this option when you want to replace the existing member in the library.

NO SIGNOUT-This option is applicable only if SIGNIN/SIGNOUT is in effect for the system. NO SIGNOUT enables the element to be retrieved without signing it out; that is, if you select this option, the element is not signed out to your user ID. This enables another user to retrieve the element at the same time you are working with it.

Similarly, if you want to use an element currently signed out to another user, you can retrieve a copy of it if that user has selected the NO SIGNOUT option.

If you use NO SIGNOUT, any CCIDS and comments are ignored. Consequently, the Master Control File is not updated

EXPAND INCLUDES-This option indicates that INCLUDE statements should be expanded when the element is copied to the external data set.

In addition, the type definition for this element must specify an INCLUDE library.

OVERRIDE SIGNOUT-If the element has been signed out to a person other than yourself, you must code this option in order to perform this action. Use OVERRIDE SIGNOUT with caution to avoid regressing changes made by another user.

SEARCH/NOSEARCH-The SEARCH option tells CA Endevor SCM to look for the element to be retrieved along the map, if it is not in the current environment. The default is SEARCH.

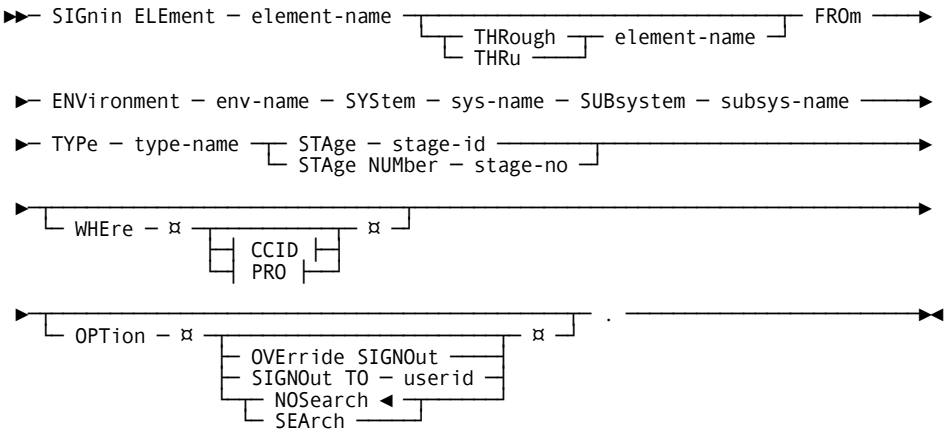
Code NOSEARCH to restrict CA Endevor SCM's search to the current environment.

Example: Retrieve SCL

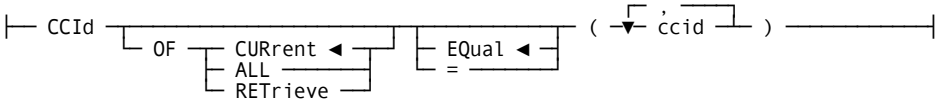
This SCL retrieves Payroll program "PAYRPT23." The map will be searched if the program is not found at Stage 1.

```
RETRIEVE ELEMENT 'PAYRPT23'
  FROM ENVIRONMENT 'PROD'
       SYSTEM 'PAYROLL'
       SUBSYSTEM 'REPORTS'
       TYPE 'COBOL'
       STAGE 1
  TO   DSNAME 'PAYROLL.SRCLIB' MEMBER 'PAYRPT31'
  OPTIONS CCID REQ#43_24
       COMMENT 'RETRIEVE THE FICA TAX REPORTING PROGRAM'
       SEARCH

       REPLACE MEMBER .
```

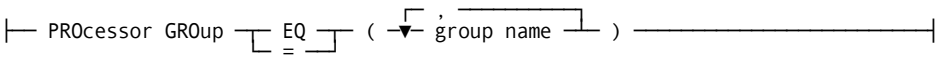
Signin Statement



Expansion of CCID



Expansion of PRO



SIGNIN ELEMENT element-name

Indicates the element(s) to be signed in. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name.

THROUGH (THRU) element-name

Indicates that a range of elements should be signed in, beginning with the element coded in the SIGNIN ELEMENT statement, up to and including the element specified in this statement. You can use a name-mask with the element name.

FROM ENVIRONMENT env-name

SYSTEM system-name

SUBSYSTEM subsys-name

TYPE type-name

STAGE stage-id

STAGE NUMBER stage-no

The FROM clause indicates the location of the element being signed in. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

You must specify an environment, system, subsystem, type, and stage. The environment name must be explicit. You can use a name-mask with the system, subsystem, type, and stage. The stage specification can be either one of the following:

- STAGE ID-Enter a single alphanumeric stage identifier.
- STAGE NUMBER-Enter either **1** or **2**.

If you use a name-mask with the stage, CA Endevor SCM begins searching for the specified element(s) in Stage 1 of the current environment, and signs in the first element that matches the specified element name, regardless of its location, version or level.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID OF ccid -Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

- **CURRENT-**Tells CA Endevor SCM to look through the CCID fields in the MCF (Master Control File) to find a specified CCID(s). This is the default.
- **ALL-**Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s). If the element is sourceless, the source element delta level CCID checks are not done.
- **RETRIEVE-**Tells CA Endevor SCM to use the CCID in the Master Control File RETRIEVE CCID field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary. The following examples illustrate the use of this clause.

```
Example 1: WHERE CCID OF CURRENT (PROJ__1, PROJ__2, PROJ__4)
Example 2: WHERE CCID OF ALL (PROJ__V)
```

WHERE PROCESSOR GROUP group name-This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The following examples illustrate the use of this clause.

```
Example 1: WHERE PROCESSOR GROUP (COBVS, COBII)
Example 2: WHERE PROCESSOR GROUP (COBV)
```

OPTIONS

OPTIONS clauses allow you to further specify action requests.

OVERRIDE SIGNOUT-If the element has been signed out to a person other than yourself, you must code this option in order to perform this action. Use OVERRIDE SIGNOUT with caution to avoid regressing changes made by another user.

SIGNOUT TO-Enables you to sign out or reassign an element at either stage to another user. If you have an element signed out to your user ID, you can use this option to reassign that element to the other user.

SEARCH/NOSEARCH-The NOSEARCH option tells CA Endevor SCM to restrict its search to the current environment. The default is NOSEARCH.

Code SEARCH to tell CA Endevor SCM to look for the element to be signed in along the map, if it is not in the current environment.

Example: Signin SCL

This SCL signs in all COBOL elements that begin with “PAYRPT*” at Stage 1 and are associated with CCID REQ#39934.

```
SIGNIN ELEMENT 'PAYRPT_'
FROM ENVIRONMENT 'PROD'
SYSTEM 'PAYROLL'
SUBSYSTEM 'REPORTS'
TYPE 'COBOL'
STAGE NUMBER 1

WHERE CCID OF CURRENT = REQ#39934.
```

Transfer Statement

The TRANSFER statement transfers an element from one location to another. There are three types of transfers:

- CA Endevor SCM to CA Endevor SCM transfers elements from one CA Endevor SCM location to another.
- CA Endevor SCM to an archive data set transfers elements from CA Endevor SCM to an archive data set.
- Archive/unload data set to CA Endevor SCM transfers elements from an archive data set or an unload tape to CA Endevor SCM.

The TRANSFER action is available in batch only. If the elements have been transferred to an archive data set, the COPY, LIST, and RESTORE actions can be executed against that data set.

Sourceless elements cannot be transferred. When an attempt to transfer an explicitly specified sourceless element is encountered, the action fails. If a sourceless element is encountered during wildcard expansion, the element is skipped and a warning message is issued.

When the element being transferred has a sourceless element downstream from the source and the target location is not upstream or is being transferred downstream before the sourceless element, and no other instance of a sourced element exists upstream, while the action has the delete source option on, the action fails. This check is performed before any source management activities occur. If no source exists upstream for a sourceless element, this would make it impossible to regenerate that element since the source no longer exists.

How to Allocate Data Sets for Large Elements

When you use the C1BM3000 utility to Transfer or Restore an element, the data sets are dynamically allocated to process data. The dynamically allocated data sets may not be sufficient, if a large element is transferred, and the System abort B37 error may occur. In this case, you can use the C1WORK01—C1WORK06 DD statements to override the standard dynamic allocation of data sets, and run the Transfer or Restore action again. These DD statements override the following records during the Transfer or Restore actions:

- C1WORK01—Merged configuration records
- C1WORK02—Configuration delta records
- C1WORK03—Configuration base records
- C1WORK04—Merged element records
- C1WORK05—Element delta records
- C1WORK06—Element base records

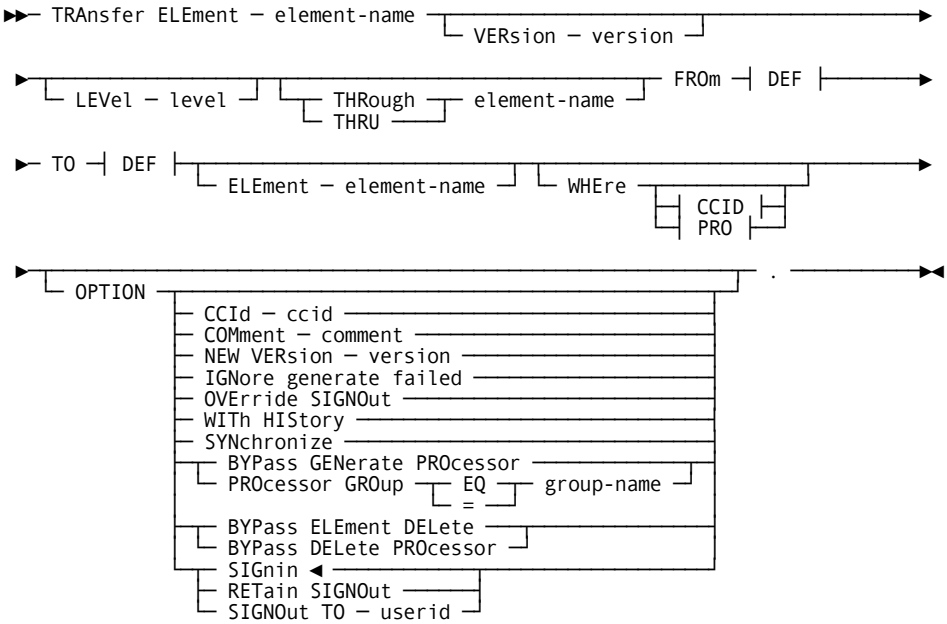
You can code any or all of the previous DD statements. If you code one C1WORK DD statement, then it overrides the equivalent dynamic allocation of datasets. For example, if a very large element is transferred the regular dynamic allocation of datasets may not be sufficient for storing element base, element delta, and merger element records. In this case, you can code the C1WORK04, C1WORK05 and C1WORK06 statements to override the dynamic allocation of datasets.

A sample JCL illustrating how to code these DD statements is shown as follows:

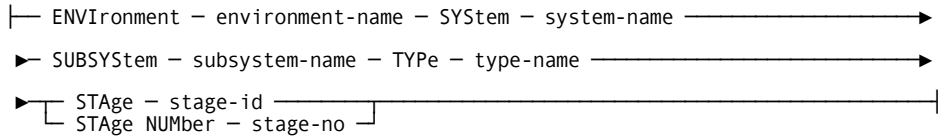
```
//STEP1     EXEC PGM=C1BM3000,PARM='SCLIN,MSGOUT1,,MSGOUT2'
//MSGOUT1   DD  SYSOUT=A
//MSGOUT2   DD  SYSOUT=A
//SCLIN     DD  *
//ARC       DD  DISP=OLD,DSN=ARCHIVE.DATASET
//C1WORK04  DD  UNIT=SYSDA,SPACE=(CYL,(50,50)),
//              DCB=(RECFM=VB,LRECL=5000,BLKSIZE=6233)
//C1WORK05  DD  UNIT=SYSDA,SPACE=(CYL,(50,50)),
//              DCB=(RECFM=VB,LRECL=5000,BLKSIZE=6233)
//C1WORK06  DD  UNIT=SYSDA,SPACE=(CYL,(50,50)),
//              DCB=(RECFM=VB,LRECL=5000,BLKSIZE=6233)
//BSTIPT01  DD  *
```

Transfer from CA Endevor SCM to CA Endevor SCM Statement

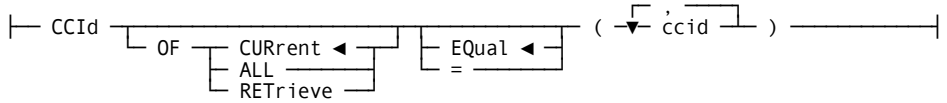
The TRANSFER FROM CA Endevor SCM TO CA Endevor SCM statement transfers elements from one CA Endevor SCM location to another.



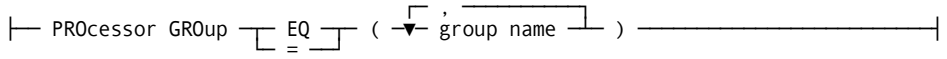
Expansion of DEF



Expansion of CCID



Expansion of PRO



TRANSFER ELEMENT element-name

Identifies the elements to be transferred. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name.

VERSION version LEVEL level

Indicates the version and level number you want to transfer. If you want to indicate a version and level, code a full element name. If you use the THROUGH clause, you cannot enter this clause i Acceptable Version numbers are 1-99. Acceptable Level numbers are 00-99. VERSION and LEVEL must be specified together.

If you do not specify a VERSION LEVEL clause, the TRANSFER action transfers all levels to the target location. If you specify this clause, only the VERSION LEVEL you indicate gets transferred.

If the specified VERSION LEVEL is not the current level, the generate processor executes at the target location, regardless of the processor group definition setting.

THROUGH (THRU) element-name

Indicates a range of elements for transfer. The range begins with the element coded in the TRANSFER ELEMENT statement, up to and including the element specified in this statement. You can use a name-mask with the element name.

FROM ENVIRONMENT env-name
SYSTEM system-name
SUBSYSTEM subsys-name
TYPE type-name
STAGE stage-id
STAGE NUMBER stage-no

The FROM clause indicates the location of the element being transferred. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

Specify an environment, system, subsystem, type, and stage. The environment name must be explicit. You can use a name-mask with the system, subsystem, type, and stage. The stage specification can be either one of the following:

- STAGE ID-Enter a single alphanumeric stage identifier.
- STAGE NUMBER-Enter either **1** or **2**.

```
TO      ENVIRONMENT env-name
        SYSTEM system-name
        SUBSYSTEM subsys-name
        TYPE type-name
        STAGE stage-id
        STAGE NUMBER stage-no
        ELEMENT element-name
```

The TO clause indicates where the element is being transferred. CA Endevor SCM uses both the TO clause in an action and any preceding SET TO clause to determine the "to" criteria for that action.

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.
- If no SET TO clause has been coded or is fully wildcarded and the TO clause system, subsystem, type, or element fields are not coded or are fully wildcarded, these fields default to the corresponding values coded in the FROM clause.

The stage specification can be either one of the following:

- STAGE ID—Enter a single alphanumeric stage identifier.
- STAGE NUMBER—Enter either **1** or **2**.

Enter a different element name if you want to change the element name specified in the TRANSFER ELEMENT clause. If you do not enter an element name here, CA Endevor SCM assigns the FROM location element name.

- You can enter a new element name only if a full element name was coded in the TRANSFER ELEMENT clause; that is, if you have not used a name-mask.
- If you want to code a different element name, you must do so in the TRANSFER statement; the SET TO MEMBER clause does not apply to this action.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID OF ccid -Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

- **CURRENT-**Tells CA Endevor SCM to look through the CCID fields in the MCF (Master Control File) to find a specified CCID(s). This is the default.
- **ALL-**Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s).
- **RETRIEVE-**Tells CA Endevor SCM to use the CCID in the Master Control File's RETRIEVE CCID field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The following examples illustrate the use of this clause.

```
Example 1: WHERE CCID OF CURRENT (PROJ__1, PROJ__2, PROJ__4)
Example 2: WHERE CCID OF ALL (PROJ___)
```

WHERE PROCESSOR GROUP *group name***-**This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The following examples illustrate the use of this clause.

```
Example 1: WHERE PROCESSOR GROUP (COBVS, COBII)
Example 2: WHERE PROCESSOR GROUP (COB_)
```

OPTIONS

OPTIONS clauses allow you to further specify action requests.

CCID ccid/COMMENT comment-You can enter a 1- to 12-character CCID and/or a 1- to 40-character comment.

CCIDs and/or comments may be required. If you do not provide a required CCID and/or comment, the TRANSFER action fails.

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

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

- Set the generate and component list delta CCID and/or COMMENT fields if the generate processor is run.
- Set the last action CCID and/or COMMENT fields.

CA Endevor SCM also:

- Clears the retrieve CCID and/or COMMENT fields.
- Sets the current source CCID and /or COMMENT fields from their value in the previous stage.
- Sets the source delta CCID and/or COMMENT fields from their last delta value in the previous stage.

When you specify a CCID and/or comment in a TRANSFER action using the WITH HISTORY option, CA Endevor SCM uses the CCID and/or comment to:

- Set the generate and component list delta CCID and/or COMMENT fields if the generate processor is run.
- Set the last action CCID and/or COMMENT fields.

CA Endevor SCM also:

- Clears the retrieve CCID and/or COMMENT fields.
- Sets the current source CCID and/or COMMENT fields from their value in the previous stage.
- Moves source delta CCIDs and COMMENTs with their respective delta levels.

When you specify a CCID and/or comment in a TRANSFER action using the SYNCHRONIZE option, CA Endevor SCM uses this CCID and/or comment to:

- Clear the retrieve CCID and/or COMMENT fields.
- Set the current source CCID and/or COMMENT fields from their value in the previous stage.
- Set the source delta CCID and/or COMMENT fields from their value at the target of the transfer, with a sync flag.

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

NEW VERSION version-By default, the version number of the FROM location element-at the time it is transferred-is assigned to the TO location element. Use this option to assign a different version number to the TO location element; simply enter the number (**1-99)** inclusive, leading zeros optional) that you want to use.

CA Endevor SCM allows only one version of an element at each location. Therefore, if the element currently exists at the target location, you cannot update it with another version. For example, if you try to transfer Version 2 of an element to a target location that already has an existing Version 1, you must archive or delete the current Version 1 before you transfer the Version 2. If the element exists at the target location or at a location up the mapped route, the TRANSFER action fails.

IGNORE GENERATE FAILED-This option applies to the *FAILED* flag previously set for the element. If the TRANSFER action is unsuccessful, you receive a message indicating that "the generate failed." Processing for the action normally is terminated at this point.

If you enter this option, however, you can perform the action whether or not the element was previously generated or moved successfully.

BYPASS GENERATE PROCESSOR-Select this option if you do not want the generate/move processor (depending on the processor group option chosen) executed for the element.

PROCESSOR GROUP EQ/= group name-Select this option to specify a predefined group of processors. If you do not specify a processor group, CA Endevor SCM defaults to the processor group last used for this element. However, the keywords PROCESSOR GROUP EQ and BYPASS GENERATE PROCESSOR are mutually exclusive (you can code one or the other). If both are coded, the default BYPASS GENERATE PROCESSOR is used.

If the FROM element is associated with a processor group that does not specify BYPASS GENERATE PROCESSOR, the processor group may be overridden with the processor group clause. Otherwise, a message will be issued saying that the processor group cannot be overridden.

OVERRIDE SIGNOUT-If the element has been signed out to a person other than yourself, you must code this option in order to perform the this action. Use OVERRIDE SIGNOUT with caution to avoid regressing changes made by another user.

BYPASS ELEMENT DELETE-This option tells CA Endevor SCM to retain the element in the FROM location after it is transferred. When you select this option, the delete processor is also bypassed.

BYPASS DELETE PROCESSOR-If you select this option, CA Endevor SCM does not execute the delete processor.

WITH HISTORY-The WITH HISTORY option preserves source element change history. If you request TRANSFER WITH HISTORY, CA Endevor SCM first ensures that the current level of the target element is the same as the base level of the source element. It then transfers all levels of the element from source to target, appending the source change history to the target change history.

If you do not code this option, CA Endevor SCM transfers the element(s) without history. When you transfer the element without history CA Endevor SCM searches through the element levels at the source location to find a matching level at the target location. CA Endevor SCM then compares the two and creates a new level at the target location that reflects the differences.

If the base level of the source element differs from the current level at the target, the TRANSFER fails unless you code the SYNCHRONIZE option.

SYNCHRONIZE- The SYNCHRONIZE option compensates for differences between the base level of a source element and the current level of a target element. CA Endevor SCM attempts to find a sync level between the source and target elements beginning with the first level of the source and works forward through the deltas. If CA Endevor SCM finds a sync level, it compares the two and creates a new level at the target that reflects the differences. If CA Endevor SCM cannot find a sync level and you specify SYNC, CA Endevor SCM issues an out of sync message. CA Endevor SCM then compares the last level of the source and last level of the target, and creates a new level at the target that reflects the differences. When moving with history, if the sync point is found, CA Endevor SCM moves the element from the FROM location to the TO location, appending the FROM location delta levels after the sync-point element. If the two levels are different, and SYNC is specified, CA Endevor SCM first creates a sync level at the target reflecting the differences between the base level of the FROM element and the target , then moves the element to the TO location and appends the FROM location delta levels to the target.

SIGNIN-This option tells CA Endevor SCM to sign in all elements at the target stage after successfully completing the move. Use this option to override SET OPTION RETAIN SIGNOUT or SET OPTION SIGNOUT TO clauses.

RETAIN SIGNOUT-This option tells CA Endevor SCM to retain the source location signouts for all elements at the target location. This option applies only if the element was signed out at the source before the TRANSFER.

- If the element was signed out at the source before the TRANSFER, it will be signed out to that same ID-at the target-after the TRANSFER.
- If the element was not signed out at the source before the TRANSFER, it will not be signed out at the target after the TRANSFER.
- If you do not use this option, the element at the target location is not signed out, regardless of whether it was signed out at the target before the TRANSFER took place.

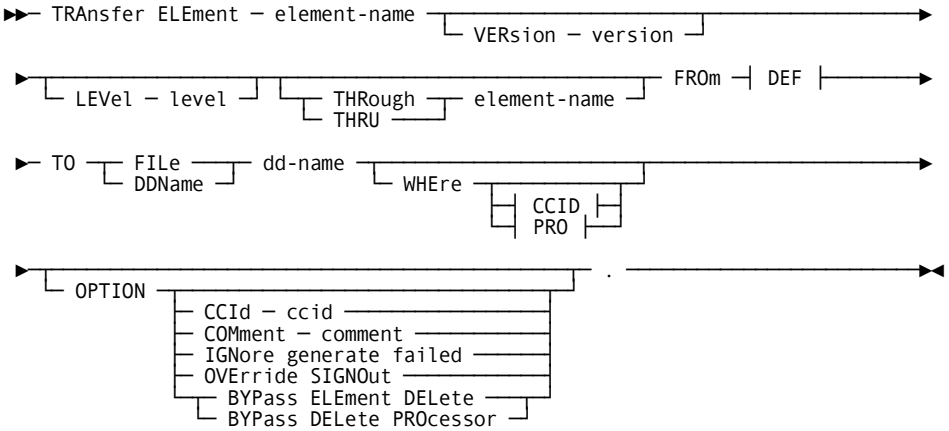
SIGNOUT TO userid-This option tells CA Endevor SCM to sign all elements out to the specified user ID at the target stage.

Transfer from CA Endevor SCM to Archive Data Set Statement

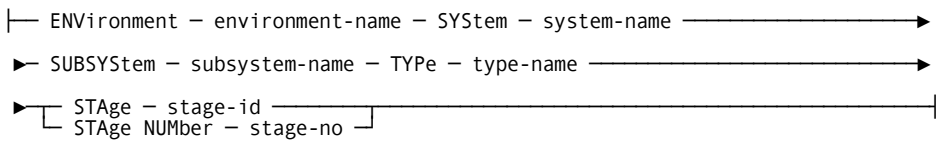
The TRANSFER FROM CA Endevor SCM TO ARCHIVE DATA SET statement transfers elements from CA Endevor SCM to an archive data set.

Sourceless elements cannot be transferred to an archive data set. When an attempt to transfer an explicitly specified sourceless element is encountered, the action fails. If a sourceless element is encountered during wildcard expansion, the element is skipped and a warning message is issued.

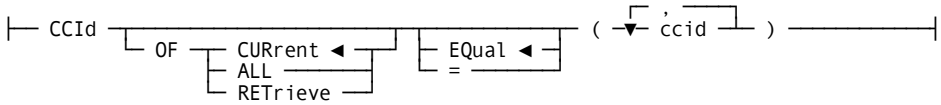
When the action has the delete source option on, it fails if the action's source element has a sourceless element downstream and no sourced element upstream from itself. The action fails before any source management activities.



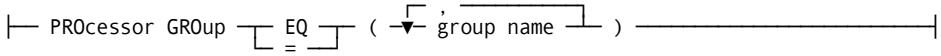
Expansion of DEF



Expansion of CCID



Expansion of PRO



TRANSFER ELEMENT element-name

Indicates the element(s) to be transferred. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name.

VERSION

Identifies the version (1-99) of the element you want to transfer. If you use this clause you must specify a full element name.

LEVEL

Identifies the level (00-99) of the element you want to transfer. If you use the LEVEL clause you:

- Cannot use the THROUGH clause.
- Must specify a full element name.

If you do not specify a LEVEL clause, the Transfer action transfers all levels to the target location. If you specify this clause, CA Endevor SCM only transfers the level you indicate.

If the specified level is not the current level, the execution of the generate processor at the target location is forced, regardless of the setting specified by the processor group definition.

THROUGH (THRU) element-name

Indicates that a range of elements should be transferred, beginning with the element coded in the TRANSFER ELEMENT statement, up to and including the element specified in this statement. You can use a name-mask with the element name.

FROM ENVIRONMENT env-name

SYSTEM system-name

SUBSYSTEM subsys-name

TYPE type-name

STAGE stage-id

STAGE NUMBER stage-no

The FROM clause indicates the location of the element being transferred. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

Note: For more information, see the description of the SET FROM statement in the chapter "Using Set, Clear, and EOF Statements (see page 49)."

You must specify an environment, system, subsystem, type, and stage. The environment name must be explicit. You can use a name-mask with the system, subsystem, type, and stage. The stage specification can be either one of the following:

- STAGE ID—Enter a single alphanumeric stage identifier.
- STAGE NUMBER—Enter either 1 or 2.

TO FILE (DDNAME) dd-name

The TO clause indicates the file or DDname to which the element is being transferred. CA Endevor SCM uses both the TO clause in an action and any preceding SET TO clause to determine the "to" criteria for that action.

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.

Note: For more information, see the description of the SET TO statement in the chapter "Using Set, Clear, and EOF Statements (see page 49)."

Note: The DCB must specify variable blocked records (RECFM=VB), and the DSORG should be PS. The minimum LRECL should be 2940 or the TYPE-LENGTH plus (+) 14, whichever is greater. When archiving to disk, the recommended block size is one-half a track and the recommended LRECL is one-half a track minus (-) 4 unless the previous rule requires a bigger LRECL/BLKSIZE. When archiving to tape, the recommended block size is 32760 and the recommended LRECL is 32756.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID OF ccid —Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

- CURRENT—Tells CA Endevor SCM to look through the CCID fields in the MCF (Master Control File) to find a specified CCID(s). This is the default.
- ALL—Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s).
- RETRIEVE—Tells CA Endevor SCM to use the CCID in the Master Control File RETRIEVE CCID field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1: WHERE CCID OF CURRENT (PROJV)
```

WHERE PROCESSOR GROUP group name—This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name.

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
Example 1: WHERE PROCESSOR GROUP (COBVS, COBII)
Example 2: WHERE PROCESSOR GROUP (COBV)
```

OPTIONS

OPTIONS clauses allow you to further specify action requests.

CCID ccid/COMMENT comment—You can enter a 1- to 12-character CCID and/or a 1- to 40-character comment. CCIDs and/or comments may be required. If you do not provide a required CCID and/or comment, the TRANSFER action fails.

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

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

- Set the generate and component list delta CCID and/or COMMENT fields if the generate processor is run.
- Set the last action CCID and/or COMMENT fields.
- CA Endevor SCM also:
  - Clears the retrieve CCID and/or COMMENT fields.
  - Sets the source CCID and/or COMMENT fields from their value in the previous stage.
  - Sets the source delta CCID and/or COMMENT fields from their last delta value in the previous stage.

When you specify a CCID and/or comment in a TRANSFER action using the WITH HISTORY option, CA Endevor SCM uses this CCID and/or comment to:

- Set the generate and component list delta CCID and/or COMMENT fields if the generate processor is run.
- Set the last action CCID and/or COMMENT fields.
- CA Endevor SCM also:
  - Clears the retrieve CCID and/or COMMENT fields.
  - Sets the source CCID and/or COMMENT fields from theirvalue in the previous stage.
  - Moves source delta CCIDs and COMMENTS with their respective delta levels.

When you specify a CCID and/or comment in a TRANSFER action using the SYNCHRONIZE option, CA Endevor SCM uses this CCID and/or comment to:

- Set the retrieve CCID and/or COMMENT fields.
- Set the source CCID and/or COMMENT fields from their value in the previous stage.
- Set the source delta CCID and/or COMMENT fields from their value at the target of the TRANSFER with a sync flag.

BYPASS ELEMENT DELETE—This option tells CA Endevor SCM to retain the element in the FROM location after it is transferred. When you select this option, the delete processor is also bypassed.

BYPASS DELETE PROCESSOR—If you select this option, CA Endevor SCM does not execute the delete processor.

IGNORE GENERATE FAILED—This option applies to the *FAILED* flag previously set for the element. If the TRANSFER action is unsuccessful, you receive a message indicating that "the generate failed." Processing for the action normally is terminated at this point.

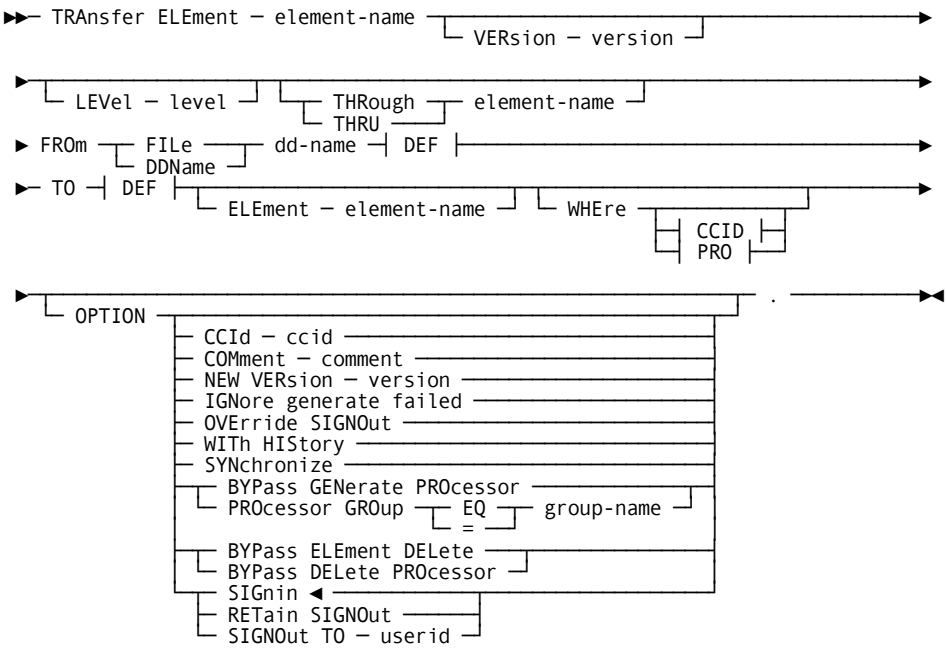
If you enter this option, however, you can perform the action whether or not the element was previously generated or moved successfully.

OVERRIDE SIGNOUT—If the element has been signed out to a person other than yourself, you must code this option in order to perform this action. Use OVERRIDE SIGNOUT with caution to avoid regressing changes made by another user.

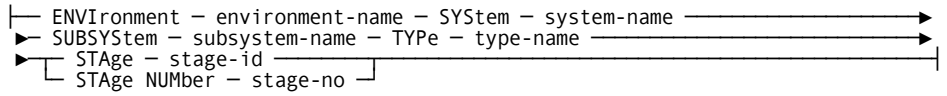
Transfer From Archive Data Set or Unload Tape to CA Endevor SCM Statement

The ARCHIVE/UNLOAD DATA SET TO CA Endevor SCM statement transfers elements from an archive data set or an unload tape to CA Endevor SCM.

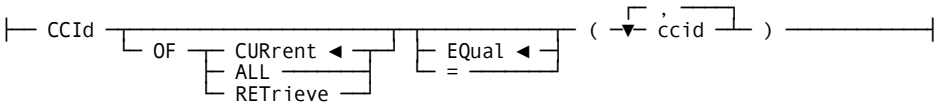
Sourceless elements cannot be transferred in from an unload tape.



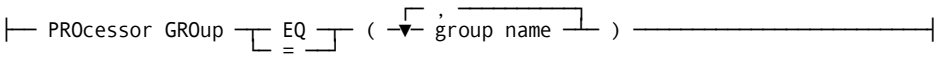
Expansion of DEF



Expansion of CCID



Expansion of PRO



TRANSFER ELEMENT *element-name*

Indicates the element(s) to be transferred. Code the required syntax and enter the appropriate element name. In addition, you can use a name-mask with the element name.

VERSION

Identifies the version (1-99) of the element you want to transfer. If you use this clause you must specify a full element name.

LEVEL

Identifies the level (00-99) of the element you want to transfer. If you use the LEVEL clause you:

- Cannot use the THROUGH clause.
- Must specify a full element name.

If you do not specify a LEVEL clause, the Transfer action transfers all levels to the target location. If you specify this clause, CA Endevor SCM only transfers the level you indicate.

If the specified level is not the current level, the execution of the generate processor at the target location is forced, regardless of the setting specified by the processor group definition.

THROUGH (THRU) *element-name*

Indicates that a range of elements should be transferred, beginning with the element coded in the RANSFER ELEMENT statement, up to and including the element specified in this statement. You can use name ask with the element name.

FROM **FILE (DDNAME)** ***dd-name***

ENVIRONMENT ***env-name***

SYSTEM ***system-name***

SUBSYSTEM ***subsys-name***

TYPE ***type-name***

STAGE NUMBER ***stage-no***

The FROM clause indicates the location of the element to be transferred. CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that ction.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

You must code a FILE or DDNAME for this request, indicating the archive data set from which the element is being transferred. Enter this information first when coding the syntax.

You must specify an environment, system, subsystem, type, and stage number (either 1 or 2). The environment name must be explicit. You can use a name-mask with the system, subsystem, and type names, as well as the stage number.

Entering a site ID is optional. This field further defines the location of the element being transferred.

TO **ENVIRONMENT** ***env-name***

SYSTEM ***system-name***

SUBSYSTEM ***subsys-name***

TYPE ***type-name***

STAGE ***stage-id***

STAGE NUMBER ***stage-no***

ELEMENT ***element-name***

The TO clause indicates the CA Endevor SCM location to which the element is being transferred. CA Endevor SCM uses both the TO clause in an action and any preceding SET TO clause to determine the "to" criteria for that action

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.

If no SET TO clause has been coded or is fully wildcarded and the TO clause system, subsystem, type, or element fields are not coded or are fully wildcarded, these fields will default to the corresponding values coded in the FROM clause.

Note: The target environment and stage values must be explicitly coded in the TO clause or SET TO clause. Partial wildcarding and name-masking are not allowed for any of the TO clause fields.

You must specify an environment, system, subsystem, type, and stage. The stage specification can be either one of the following:

- STAGE ID—Enter a single alphanumeric stage identifier.
- STAGE NUMBER—Enter either 1 or 2.

Remember that you cannot use a name-mask with a TO field location.Enter a different element name if you want to change the element name specified in the TRANSFER ELEMENT clause. If you do not enter an element name here, CA Endevor SCM uses the archived element name.

- You can enter a new element name only if a full element name was coded in the TRANSFER ELEMENT clause; that is, if you have not used a name-mask.
- If you want to code a different element name, you must do so in the TRANSFER statement; the SET TO MEMBER clause does not apply to this action.

WHERE

Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID *ccid*

Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

The next example illustrates the use of this clause.

```
WHERE CCID EQ PROJ4)
```

WHERE ARCHIVE

This clause allows you to select elements based on the date and, optionally, time that an element was archived. There are four possible forms for this clause:

WHERE ARCHIVE DATE *mm/dd/yy* [TIME *hh:mm*]

This clause tells CA Endevor SCM to archive only those elements with this date, and optionally, this time stamp.

WHERE ARCHIVE FROM DATE *mm/dd/yy* [TIME *hh:mm*]

This clause tells CA Endevor SCM to archive all elements with a date and, optionally, a time stamp on or after the specified date and time stamps.

WHERE ARCHIVE THROUGH DATE *mm/dd/yy* [TIME *hh:mm*]

This clause tells CA Endevor SCM to archive all elements with a date and, optionally, a time stamp earlier than and including the specified date and time stamp.

WHERE ARCHIVE FROM DATE *mm/dd/yy* [TIME *hh:mm*] THROUGH DATE *mm/dd/yy* [TIME *hh:mm*]

This clause tells CA Endevor SCM to archive only those elements with a date, and optionally, a time stamp within the specified range. If you enter a time, you must enter the date with it.

WHERE PROCESSOR GROUP *group name*

This clause is not valid when transferring from an archive data set. This clause allows you to select elements according to a specified processor group. You can use a name-mask when specifying the processor group name

If you need to select elements identified under more than one processor group, you can specify multiple distinct processor group selectors by enclosing the processor groups with parentheses and separating them with commas. The processor groups may extend over multiple lines if necessary.

The next examples illustrate the use of this clause.

```
WHERE PROCESSOR GROUP (COBVS, COBII)
```

```
WHERE PROCESSOR GROUP (COBV)
```

OPTIONS

OPTIONS clauses allow you to further specify action requests.

CCID *ccid*/COMMENT *comment*

You can enter a 1- to 12-character CCID and/or a 1- to 40-character comment.

CCIDs and/or comments may be required. If you do not provide a required CCID and/or comment, the TRANSFER action fails.

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

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

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

- Set the generate and component list delta CCID and/or COMMENT fields if the generate processor is run.
- Set the last action CCID and/or COMMENT fields.
- CA Endevor SCM also:
- Clears the retrieve CCID and/or COMMENT fields.
- Sets the source CCID and /or COMMENT fields from their value in the previous stage.
- Sets the source delta CCID and/or COMMENT fields from their last delta value in the previous stage.

When you specify a CCID and/or comment in a TRANSFER action using the WITH HISTORY option, CA Endevor SCM uses the CCID and/or comment to:

- Set the generate and component list delta CCID and/or COMMENT fields if the generate processor is run.
- Set the last action CCID and/or COMMENT fields.
- CA Endevor SCM also:
- Clears the retrieve CCID and/or COMMENT fields.
- Sets the source CCID and/or COMMENT fields from their value in the previous stage.
- Moves source delta CCIDs and COMMENTs with their respective delta levels.

When you specify a CCID and/or comment in a TRANSFER actionusing the SYNCHRONIZE option, CA Endevor SCM uses this CCID and/or comment to:

- Set the retrieve CCID and/or COMMENT fields.
- Set the source CCID and/or COMMENT fields from their value in the previous stage.
- Set the source delta CCID and/or COMMENT fields from their value at the target of the transfer, with a sync flag.

NEW VERSION *version*

Use this option to assign a different version number to the TO location element. Acceptable values are 1-99. CA Endevor SCM allows only one version of an element at each location. For example, if you try to transfer Version 2 of an element to a target location that already has an existing Version 1, you must archive or delete the current Version 1 before you transfer the Version 2. If the element exists at the target location or at a location up the mapped route, the TRANSFER action fails.

OVERRIDE SIGNOUT

If the element is signed out to another person, you must code this option in order to perform this action. Use OVERRIDE SIGNOUT with caution to avoid regressing changes made by another user.

BYPASS GENERATE PROCESSOR

Use this option if you do not want the generate processor executed for the element. Otherwise, CA Endevor SCM looks for and executes the generate processor for the element when it is transferred.

PROCESSOR GROUP EQ/= *group name*

Select this option to specify a predefined group of processors. If you do not specify a processor group, CA Endevor SCM defaults to the processor group last used for this element. The keywords PROCESSOR GROUP EQ and BYPASS GENERATE PROCESSOR are mutually exclusive (you can code one or the other). If both are coded, the default BYPASS GENERATE PROCESSOR is used.

WITH HISTORY

The WITH HISTORY option preserves source element change history. If you request TRANSFER WITH HISTORY, CA Endevor SCM first ensures that the current level of the target element is the same as the base level of the source element. It then transfers all levels of the element from source to target, appending the source change history to the target change history.

If you do not code this option, CA Endevor SCM transfers the element without history. When you transfer the element without history CA Endevor SCM searches through the element levels at the source location to find a matching level at the target location. CA Endevor SCM then compares the two and creates a new level at the target location that reflects the differences.

If the base level of the source element differs from the current level at the target, the transfer fails unless you code the SYNCHRONIZE option.

SYNCHRONIZE

When transferring either with or without history, the SYNCHRONIZE option compensates for differences between the base level of a source element and the current level of a target element.

If these levels differ, the SYNCHRONIZE option tells CA Endevor SCM to create a new level at the target that reflects the differences. After creating the sync level, CA Endevor SCM transfers the element(s), either with or without history.

SIGNIN

This option tells CA Endevor SCM to sign in all elements at the target stage. Use this option to override SET OPTION RETAIN SIGNOUT or SET OPTION SIGNOUT TO clauses.

RETAIN SIGNOUT

This option tells CA Endevor SCM to retain the source location signouts for all elements at the target location. This option applies only if the element was signed out at the source before the TRANSFER.

- If the element was signed out at the source before the TRANSFER, it will be signed out to that same ID—at the target—after the TRANSFER.
- If the element was not signed out at the source before the TRANSFER, it will not be signed out at the target after the TRANSFER.
- If you do not use this option, the element at the target location is not signed out, regardless of whether it was signed out at the target before the TRANSFER took place.

SIGNOUT TO *userid*

This option tells CA Endevor SCM to sign all elements out to the specified user ID at the target stage.

Example of Transfer SCL

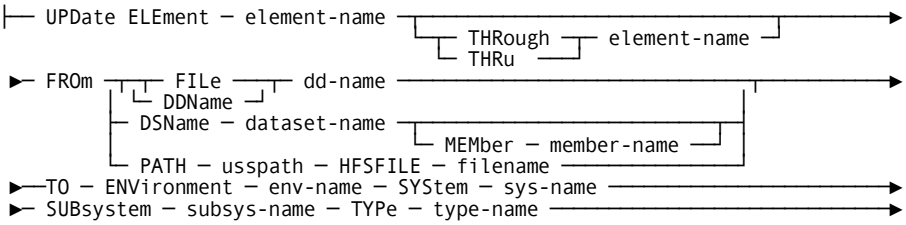
The following is an example of TRANSFER SCL. This SCL transfers all of the "PAYRPT*" COBOL elements to the NEWREPRT subsystem. All element history will be retained, and the signout status can be overridden, if necessary.

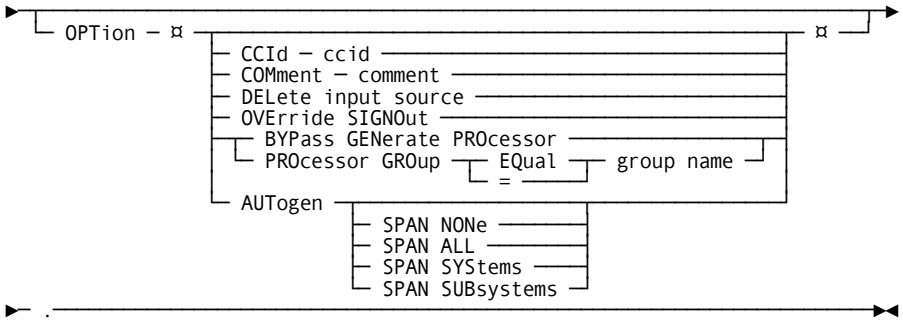
```
TRANSFER ELEMENT 'PAYRPT'
FROM ENVIRONMENT 'PROD'
SYSTEM 'PAYROLL'
SUBSYSTEM 'REPORTS'
TYPE 'COBOL'
STAGE NUMBER 1
TO ENVIRONMENT 'PROD'
SYSTEM 'PAYROLL'
SUBSYSTEM 'NEWREPRT'
TYPE 'COBOL'
STAGE NUMBER 1
OPTIONS CCID REQ#4418
COMMENT 'MOVE REPORTING SUBSYSTEM PROGRAMS'
WITHHISTORY
OVERRIDE SIGNOUT .
```

Update Statement

The UPDATE statement updates an element in the entry stage, thereby creating a new level for the element in the entry stage. The entry stage for the environment is defined through the C1DEFLTS table. Elements are updated only if there are differences between the incoming source in the FROM location and the target entry stage source.

If the element at the entry stage is a sourceless element, the sourced element from up the map is fetched back prior to the update. Before the fetch, CA Endevor SCM checks the last level timestamps between the sourceless and sourced elements. If both are the same, the action continues. If not, the action fails. If the action fails, the user needs to determine the corrective action.





UPDATE ELEMENT element-name

Indicates the element(s) to be updated. Code the required syntax and enter the appropriate element name; up to 255 characters are allowed. In addition, you can use a name-mask with the element name.

THROUGH (THRU) element-name

Indicates that a range of elements should be updated, beginning with the element coded in the UPDATE ELEMENT statement, up to and including the element specified in this statement. You can use a name-mask with the element name. If you use the THROUGH clause, however, you cannot enter a member name (in the FROM clause).

Note: If you are working with a sequential file, the THROUGH clause is ignored.

FROM FILE (DDNAME) dd-name

DSNAME dataset-name

MEMBER member-name

PATH usspath

HFSFILE filename

The FROM clause indicates the location of the element being updated. Endevor uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action.

- A FROM clause in an action overrides values in a SET FROM clause that precedes the action.
- If the SET FROM clause contains values that are not included in the FROM clause, Endevor uses these values.

For more information, see the description of the SET FROM statement, in Chapter 3, "Set Clear, and EOF Statements."

You must enter a FILE, DDNAME, DSNAME, or PATH in conjunction with the HFSFILE (enter one and only one). If you enter a FILE or DDNAME, be sure the appropriate JCL is coded. Enter a member name (up to 255 characters) if it differs from the element name specified in the UPDATE ELEMENT clause; you can use a name-mask with this entry. If you do not enter a member name, Endevor assumes that the element name and member name are the same.

- You can enter a member name only if a full element name has been coded in the UPDATE ELEMENT clause; that is, if you have not used a name-mask.
- If you want to code a member name, you must do so in the UPDATE statement; the SET FROM clause does not contain a member name entry. If you do enter a member name, you cannot enter a THROUGH clause.
- If you are working with a sequential file, the MEMBER clause is ignored

PATH

The USS directory where the element source file resides.

HFSFILE

The file in the USS directory that you want to put under the control of CA Endevor SCM.

TO **ENVIRONMENT env-name**

SYSTEM sys-name

SUBSYSTEM subsys-name

TYPE type-name

The TO clause indicates where the element is being updated. CA Endevor SCM uses both the TO clause in an action and any preceding SET TO clause to determine the "to" criteria for that action.

- A TO clause in an action overrides values in a SET TO clause that precedes the action.
- If the SET TO clause contains values that are not included in the TO clause, CA Endevor SCM uses these values.

You must specify an environment, system, subsystem, and type. Remember that you cannot use a name-mask with a TO field location.

OPTIONS

OPTIONS clauses allow you to further specify action requests.

CCID *ccid* COMMENT *comment*

You can enter a one- to twelve-character CCID, a one- to forty-character comment. CCIDs, or both. CCIDs and comments may be required. If you do not provide a required CCID or comment, the UPDATE action fails. When you specify a CCID or comment in an UPDATE action for an existing element, CA Endevor SCM uses this CCID or comment to:

- Set the source and source delta CCID or COMMENT fields, if the CCID or comment has changed.
- Set the generate CCID or COMMENT fields, if the generate processor is run.
- Set the component list delta CCID or COMMENT fields, if running the generate processor creates a change.
- Set the last action CCID or COMMENT fields.

CA Endevor SCM also clears the entry stage retrieve CCID or COMMENT fields when you update an element. If you use the BYPASS GENERATE PROCESSOR, the UPDATE action will not set the generate or component delta CCID or COMMENT fields.

DELETE INPUT SOURCE

After an element has been successfully updated in CA Endevor SCM, you can use this option to remove the member from the library in which it originated. If you input a sequential file, this option deletes that file.

OVERRIDE SIGNOUT

If the element has been signed out to a person other than yourself, you must code this option in order to perform this action. Use OVERRIDE SIGNOUT with caution to avoid regressing changes made by another user.

BYPASS GENERATE PROCESSOR

Use this option if you do not want the generate processor executed for the element. Otherwise, CA Endevor SCM looks for and executes the generate processor for the element when it is updated.

PROCESSOR GROUP group name

Use this option to specify a predefined named group of processors. If you do not specify a processor group, CA Endevor SCM defaults to the processor group last used for this element.

AUTOGEN | SPAN NONE| SPAN ALL | SPAN SYSTEMS | SPAN SUBSYSTEMS

Applicable for Add, Update, and Generate actions in batch requests. This option cannot be used in packages and the option does not work if Bypass Generate Processor is set. The Global Type Sequencing batch processing method must be enabled. Autogen only acts on components whose Types are listed in the Global Type Sequencing table. If the component's Type is not listed in the Global Type Sequencing table, the Autogen request is ignored. In addition, your site must have purchased and activated the CA Endevor Automated Configuration. Autogen can be specified alone or with various Span keyword options. "Autogen Span" is *not* a valid option. Valid options follow:

AUTOGEN

Generates all elements that use the component that is the target of the action. These *using* elements are generated at the target location that is specified in the SCL statement. If they do not exist at the target location, they are brought back to the target location as sourceless elements. An administrator can change the behavior of the Autogen feature, by activating AUTOGEN_SOURCE in the Optional Features Table (ENCOPTBL). When this option is activated, the Generate actions for the using elements are built with the Copyback, instead of the NoSource, option. For more information about sourceless elements, see the NoSource option description in Generate Syntax (see page 116).

Note: *Using* elements are elements that use the element that is the target of an Add , Update, or Generate action. For example, if Autogen is specified for copybook, COPYA, then the programs that use that copybook are known as using elements.

AUTOGEN SPAN NONE

Generates all elements that use the component being acted upon. This option has the exact same effect as the option "AUTOGEN."

AUTOGEN SPAN ALL

Generates using elements that are found in any System and Subsystem combinations within the Environment and Stage of the component's logical map.

AUTOGEN SPAN SYSTEMS

Generates using elements found in any System, provided the element's Subsystem name matches the name of the Subsystem of the target component. Only Systems found within the Environment and Stage of the component's logical map or higher up the map are searched. This option is different from the Autogen option in that it includes additional Systems with the same Subsystem name in the search.

AUTOGEN SPAN SUBSYSTEMS

Generates using elements from all Subsystems with the same-named System of the component specified. Only Subsystem found in the System of the target component within the Environment and Stage of the component's logical map or higher up the map are searched. This option is different from the Autogen option in that it includes additional Subsystems with the same System in the search.

The following restrictions apply to the SPAN options:

- Common libraries must be included in the generate processor concatenations of the Systems and Subsystems of the using elements. If the libraries are not included, then the Generate action does include the changes made to the component element when searching across Systems or Subsystems.
- The same Systems and Subsystems must exist in each Environment and Stage location. For example, if ELEMENTA is a using element in the PROD Environment, system A, then system A must exist in the DEV Environment also.
- SPAN only includes using elements from Systems or Subsystems located in the target Environment and higher up the map.
- SPAN does not include using elements from outside the Environment map.

Example: Update SCL

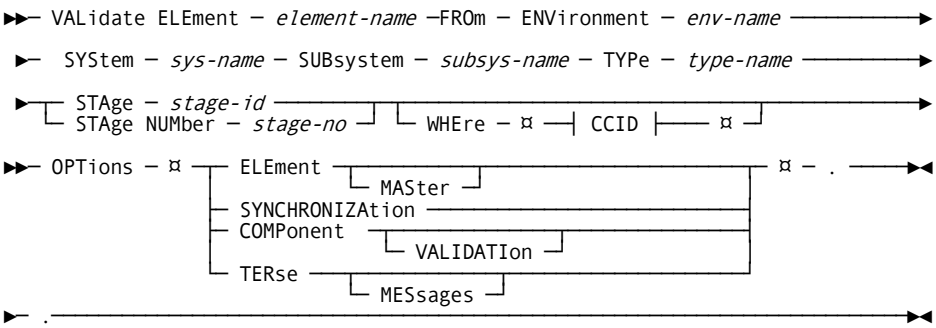
The following is an example of UPDATE SCL. This SCL modifies the Payroll Reporting program "PAYRPT23." After the update is complete, the source member will be deleted.

```
UPDATE     ELEMENT 'PAYRPT23'
  TO       ENVIRONMENT 'PROD'
             SYSTEM 'PAYROLL'
             SUBSYSTEM 'REPORTS'
             TYPE 'COBOL'
  FROM     DSNAME 'PAYROLL.SRCLIB'
  OPTIONS DELETE INPUT SOURCE
             CCID REQ#42976
             COMMENT 'CHANGES FOR NEW REPORTING REQUIREMENTS' .
```

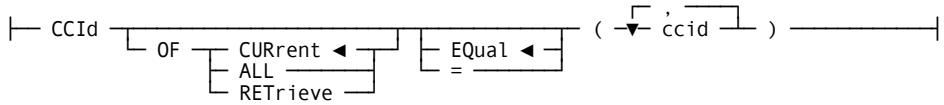
Validate Statement

The Validate statement checks to make sure that elements were generated correctly, no synchronization errors are detected, and that all the components exist and are valid. By specifying the Element Master, Component Validation, and Synchronization options in the SCL statement, you can choose to perform all or some of these Validate checks. The Validate action can be performed at any time, enabling developers to ensure the integrity of elements before including them in a package. The advantage of this feature is that you do not have to create a package and attempt to cast it in order to perform these tasks. (Casting a package also performs these actions.) The Validate command can only be specified in batch and can not be part of package SCL statements.

The Validate action supports long name elements. Long name elements are not available for ISPF dialog processing, but the long name location information can be manually coded in the SCL.



Expansion of CCID



VALidate ELEment *element-name*

Indicates the element to be validated.

FROM

The FROM clause indicates the location of the element being validated. A check is performed to ensure the element exists at the location specified in the action. If not, an E level message is issued.

You must specify an environment, system, subsystem, type, and stage. The environment name must be explicit. You can use a name-mask with the system, subsystem, type, and stage.

ENVIRONMENT *env-name*
SYSTEM *sys-name*
SUBSYSTEM *subsys-name*
TYPE *type-name*
STAGE *stage-id* | STAGE NUMBER *stage-no*

The stage specification can be either one of the following:

- STAGE -Enter a single alphanumeric stage identifier.
- STAGE NUMBER-Enter either **1** or **2**.

Note: CA Endevor SCM uses both the FROM clause in an action and any preceding SET FROM clause to determine the "from" criteria for that action. A FROM clause in an action overrides values in a SET FROM clause that precedes the action.If the SET FROM clause contains values that are not included in the FROM clause, CA Endevor SCM uses these values.

WHERE

(Optional) Use WHERE clauses to further qualify element selection criteria. CA Endevor SCM uses both the WHERE clause in an action and any preceding SET WHERE clause to determine the "where" criteria for that action.

- A WHERE clause in an action overrides values in a SET WHERE clause that precedes the action.
- If the SET WHERE clause contains values that are not included in the WHERE clause, CA Endevor SCM uses these values.

WHERE CCID OF *ccid*

Limits the processing to those elements that match one of the supplied CCIDs. You can use a name-mask in this field.

- **CURRENT-**Tells CA Endevor SCM to look through the CCID fields in the MCF (Master Control File) to find a specified CCID(s). This is the default.
- **ALL-**Tells CA Endevor SCM to search both the Master Control File and the SOURCE DELTA levels for a specified CCID(s). If you have ACM, CA Endevor SCM also searches the COMPONENT LIST DELTA levels for the specified CCID(s).
- **RETRIEVE-**Tells CA Endevor SCM to use the CCID in the Master Control File RETRIEVE CCID field.

If you need to select elements identified under more than one CCID, you can specify multiple CCIDs by enclosing the CCIDs with parentheses and separating them with commas. The CCIDs may extend over multiple lines if necessary.

Although the foreground SCL Generation panel allows you to filter the selection list by using a single CCID, the SCL supports multiple CCIDs.

The next examples illustrate the use of this clause.

```
Example 1: WHERE CCID OF CURRENT (PROJ001, PROJ002, PROJ004)
Example 2: WHERE CCID OF ALL (PROJ00V)
```

OPTIONS

OPTIONS clauses allow you to further specify action requests. The Validate Action statement requires at least one of the options Element Master, Synchronization, or Component Validation. If none of these are coded, then a warning message is issued to inform you of this situation and the action is processed assuming all three options are enabled.

ELEMENT MASTER

Checks to make sure the element is generated correctly. If this option is enabled the following checks are performed. If any of these conditions are detected, error messages are issued.

- If the processor is unknown, an error message is issued. This condition may occur when the last action is a RESTORE and the NOGEN option was specified.
- If the element is not generated or the last generate failed, an error message is issued. An exception to this rule is if the last processor is a delete processor.
- If this element is currently locked in a package, an error message is issued.

SYNCHRONIZATION

Checks to make sure that an out-of synch condition is not caused by the action being performed. If this option is enabled, the following checks are performed;

- A check is performed to determine if a second copy of this element exists up the logical map.
- If a second copy of this element exists up the logical map, the two copies of the element are used as input for the synchronization check processing. The location of the second copy of the element will be displayed in the C1MSGS messages file along with synchronization messages to indicate the results of the check.
- If the synchronization check is performed, an additional check is also performed to see if another copy of the element exists at a physical location in-between the two copies that were used as input in the synchronization checking. If a copy is found at an in-between location information messages are written to the C1MSGS report file.

The synchronization logic for this feature looks up the logical map for another copy of this element, starting at the inventory location specified in the action. It ignores other parallel maps that are defined in the C1DEFLTS table.

The synchronization logic determines if a second copy of this element resides up the logical map. If a second copy is not found, an informational message is written to the report and the synchronization logic is complete. The physical map is not searched.

COMPONENT VALIDATION

Checks to make sure that all the components exist and are valid. The CA Endevor SCM Automated Configuration option must be installed at your site in order to use the component validation option. If this option is enabled, the following checks are performed:

- The components of the element (parent) specified in the action are obtained and added to the list of elements that must be validated.
- The footprints of each component element are compared against the actual location of that element to ensure the latest copy of each component is included in the parent element and that the component exists.

Footprint error messages are issued for any component found to be in error. Totals are displayed as to how many components were verified and how many contained errors.

TERSE MESSAGES

Limits the amount of message detail that is written to the C1MSGS1 report file for those messages that result from the Validate action. All action output produced by the VALIDATE action is written to the C1MSGS1 report file. The terse messages option controls the amount of message detail that is written to that file. It will only write key I level messages and non-I level messages to the C1MSGS report file. If this option is not specified, then all messages are written to the C1MSGS report file.

Example: Validate Action Syntax with All Options

This Validate action SCL validates all the inventory located in the DEV environment PAYROLL sandbox subsystem. All available action options are specified.

```
VALIDATE ELEMENT 'PAYRPT17'
FROM ENVIRONMENT 'DEV'
SYSTEM 'FINANCE'
SUBSYSTEM 'PAYROLL'
TYPE *
STAGE NUMBER *
OPTIONS ELEMENT MASTER
SYNCHRONIZATION
COMPONENT VALIDATION
TERSE .
```

Chapter 5: Managing Packages in Batch

This section contains the following topics:

The Batch Package Facility

CA Endevor SCM's Batch Package Facility provides you with the ability to execute all package actions in batch mode. Sample JCL is supplied in CSIQJCL member member ENBP1000. In addition, the CA Endevor SCM Batch Package Facility:

- Supports all foreground package actions. For more information about package processing, see the *Packages Guide*.
- Provides the additional actions SUBMIT, ARCHIVE, and INSPECT.
- Has the same package status requirements as those used in foreground. For more information about package status requirements, see the *Packages Guide*.
- Supports before- and after-package exits.
- Invokes the GENPKGID exit, if installed, to generate a new package ID. For more information about package exits, see the *Exits Guide*.

Summary of Batch Package Actions

The following table summarizes CA Endevor SCM batch package actions, their required status and exits supported.

Action	Description	Required Status	Exits Before	Supported After
Approve Package	Approves a package for execution	In-approval Denied	X	X
Archive Package	Copies the package definitions to an external data set	Executed if backouts exist Committed if backout is enabled	X	X
Backin Package	Backs a package in, reversing the BACKOUT PACKAGE action	Executed	X	X
Backout Package	Backouts package changes. Restores output modules to pre-execution state.	Executed Exec-failed	X	X
Cast Package	Casts a package, which freezes the data and prevents further changes. For a list of cast validations, see the *Packages Guide*.	In-Edit	X	X
Commit Package	Commits a package removing all backout/backin data, but retaining package event information.	Executed	X	X
Define Package	Creates a new or updates an existing package	In-edit for an existing package	X	X
Delete Package	Deletes an entire package from CA Endevor SCM	Any status	X	X
Deny Package	Denies execution of a package	In-approval	X	X
Execute Package	Executes a package	Approved Exec-failed	X	X
Export Package	Writes the SCL associated with a package to an external data set	Any status	X	X

Action	Description	Required Status	Exits Before	Supported After
Inspect Package	The Inspect action checks each element for security, signout, and synchronization conflicts and source changes and reports on the changes in element status that might effect the successful execution of the package. For a list of Inspect validations, see the *Packages Guide*.	Approved Exec-failed		
Reset Package	Resets a package back to a status of In-edit.	Any status	X	X
Submit Package	Submits a JCL job stream to execute one or more packages	Approved Executed if the WHERE PACKAGE STATUS IS EXECFAILED clause is specified and if the package execution has previously failed		

Batch Package Actions and Wildcarding

You can use wildcard package IDs for the following package actions:

- Archive
- Cast
- Commit
- Delete
- Execute
- Submit

When you wildcard a package ID, CA Endevor SCM selects packages against which you are authorized to perform actions. It is possible that a package ID matches the wildcard you specify but is not selected for the following reasons:

- The package is in the wrong state for the action selected.
- The package is non-sharable and you are not the owner of the package.
- The package has one or more approver groups associated with it and you are not a member of any of those approver groups.

Note: SCL inside of packages may not contain any wildcards.

Batch Package Facility Execution

The Batch Package Facility, program ENBP1000, performs package actions by executing SCL statements specified in the ENPSCLIN DD statement.

Note: For more information about SCL coding conventions, see the chapter "Using SCL (see page 39)."

The following general rules apply to ENBP1000 execution:

- There is no defined limit to the number of package actions the facility can process.
- There is no defined limit to the number of SCL statements that you can specify.
- Statements are executed in the sequence provided.
- Statements are parsed before any package actions are executed.
- If parse errors are detected, none of the actions are executed.
- Actions are processed as long as the action return code is 12 or less. If a return of greater than 12 is received, all remaining actions are bypassed.
- If the same clause is specified multiple times in a statement, the last clause specified is the one used.

DD Statement Descriptions

ENPSCLIN

Defines the Batch Package Facility control statements. The DD statement can refer to instream data, a sequential data set, or a partitioned data set with an explicit member.

Where a partitioned data set option is used, only one archive action permitted. You may archive multiple packages, however it must be done in the same command. If you use a separate archive action for each package being archived, only the last package in the PDS member will appear. For example:

```
Archive Package to DDN ddname
```

Options where older than N days will archive all eligible packages to a PDS member. Whereas the SCL:

```
Archive Package A to DDN ddname
Archive Package B to DDN ddname
```

will result in only package B residing in the Archive file.

The archive package file must be defined with variable length records and have a minimum record length of 4200. The data set blocksize must be 4 bytes greater than the record length. We recommend that you define a blocksize of 0 and let the system default to the optium block size.

If any of the data set attributes are incorrect, an error message is written and a return code 12 is set.

C1MSGS1

Defines the destination of the Batch Package Facility execution reports. You can write the Batch Package Facility Summary report to a different location by uncommenting the C1MSGS2 DD statement in the sample JCL.

JCLIN

Identifies the default location of the JCL jobcard to be used by the SUBMIT PACKAGE action. The data set can be a sequential data set or a partitioned data set with an explicit member. The DD statement is used only with the SUBMIT PACKAGE action.

JCLOUT

Identifies the default output of the SUBMIT PACKAGE action. Generally, the DD statement refers to an internal reader but it can also refer to a sequential data set or a partitioned data set with a explicit member name. The DD statement is used only with the SUBMIT PACKAGE action.

Validating Input SCL

You can check the syntax of your SCL statements before submitting them for execution by using an optional parameter of VALIDATE on the JCL PARM statement. When you specify the VALIDATE parameter, the statements in the ENPSCLIN DD statement are parsed. The statements are not executed. To specify the VALIDATE parameter, change the PARM= statement on the sample JCL to PARM='ENBP1000VALIDATE'.

Batch Package Facility Return Codes

The Batch Package Facility passes one of the following return codes after execution is complete:

0

All actions were performed successfully.

4

One or more actions completed with a warning message.

8

One or more actions completed with a caution message.

12

One or more action completed with an error message. The action may not have completed successfully.

16

An unrecoverable error occurred.

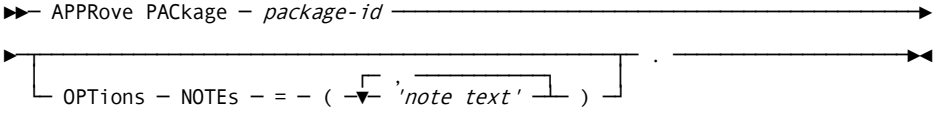
20

The C1MSGS1 DD statement was not allocated or the C1MSGS1 file could not be initialized.

The Approve Package Action

The APPROVE PACKAGE action approves packages for execution. Use the APPROVE PACKAGE action against a package only if the package has a status of In-approval or Denied.

Approve Package Syntax



APPROVE PACKAGE package-id

The APPROVE PACKAGE clause identifies the package you are approving. You must use a fully specified package ID. The package ID can include imbedded spaces. If the package ID contains an imbedded space or if the ID comprises only numeric characters (for example, 12345), enclose the package ID in either single or double quotation marks.

OPTIONS

OPTION clauses allow you to further specify package actions.

NOTES

Use the NOTES clause to add remarks to the package definition. Enclose the note text in either single or double quotation marks. If you use multiple text lines, enclose each text line in quotation marks and separate by commas. You can specify up to 8 note text lines of up to 60 characters each. This text replaces any text that is already associated with the package.

Example: Approve Package SCL

The following is an example of APPROVE PACKAGE SCL. The SCL approves the package called PAYROLLPKGO1.

```
APPROVE PACKAGE PAYROLLPKG01.
```

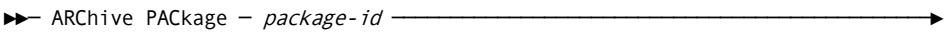
The Archive Package Action

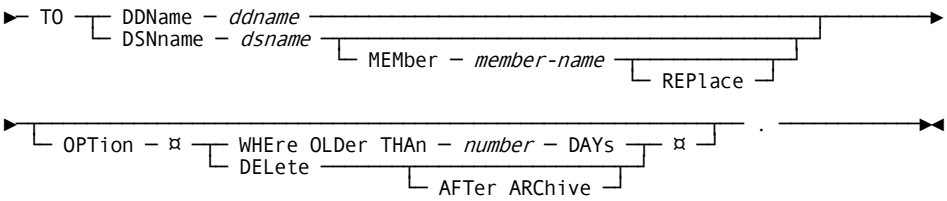
The ARCHIVE PACKAGE action offloads a package definition to an external data set. The ARCHIVE PACKAGE action can, optionally, delete the package after it is successfully written to the external data set. The ARCHIVE PACKAGE action archives the current version of a package. For promotion packages, the current version and all historical versions are archived.

You can use the ARCHIVE action against a package that has a status of Executed or against a package that has a status of Committed. If the status is Executed, you cannot use the ARCHIVE action with the delete option against any package that has backout members.

Note: When you archive a package to a partitioned data set (PDS), you can generate reports only from individual members of the data set and not the complete data set. For more information, see the *Reports Guide*.

Archive Package Syntax





ARCHIVE PACKAGE *package-id*

The ARCHIVE PACKAGE clause identifies the package you are archiving. You can either fully specify, partially wildcard or fully wildcard the package ID. If you wildcard the package ID and specify the OPTIONS DELETE AFTER ARCHIVE clause, you must specify the WHERE OLDER THAN clause. If you fully specify the package ID, the WHERE OLDER THAN clause is ignored.

You can include imbedded spaces in the package ID. If the package ID contains an imbedded space or comprises only numeric characters (for example, 12345) then enclose the package ID in either single or double quotation marks.

TO **DDNAME ddname**

DSNAME dsname

MEMBER member-name

REPLACE

The TO clause identifies the data set to which you are archiving the package. You must enter either a DDname or a data set name. Specify only one of the two statements. The archive package file must be defined with variable length records and have a minimum record length of 4200. The data set blocksize must be 4 bytes greater than the record length. We recommend that you define a blocksize of 0 and let the system default to the optimum block size

The TO DDNAME clause identifies the name of an allocated DD statement. The DD statement must reference a sequential data set or a partitioned data set with an explicit member.

Note: When coding mulitple ARCHIVE PACKAGE statements with the TO DDNAME clause, use a DISP=MOD to assure that all data is retained.

The TO DSNAME clause identifies the name of an existing, catalogued data set. If the data set is a partitioned data set, you can use the member clause to specify a member name to be created. If you do not specify a MEMBER clause, the member name created is TEMPNAME. The REPLACE clause replaces an existing, like-named member. You can use the REPLACE clause is only if the MEMBER clause is specified.

OPTIONS

OPTION clauses allow you to further specify package actions.

WHERE OLDER THAN number DAYS-This clause allows you to specify the minimum age of the package you are archiving. The package must be older than the number of days you specify in order for it to be archived. For example, if you specify WHERE OLDER THAN 30 DAYS and the current date is January 31, only packages executed successfully or committed on or before January 1 are archived. There is no default value for the WHERE OLDER THAN clause. You must specify the WHERE OLDER THAN clause if you wildcard the package ID and specify the OPTIONS DELETE AFTER ARCHIVE clause. If you fully specify the package ID, the WHERE OLDER THAN clause is ignored. The WHERE OLDER THAN value must be between 0 and 999, inclusive. You receive an error message if you specify a value outside this range.

DELETE AFTER ARCHIVE-This option deletes the package after the package definitions are successfully archived. If this is a promotion package, all versions of the package are deleted.

Example : Archive Package SCL

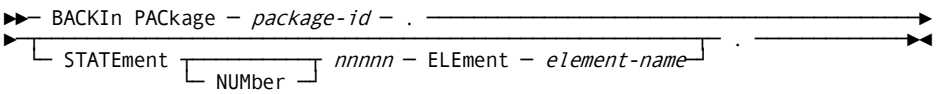
The following is an example of ARCHIVE PACKAGE SCL. The SCL archives all packages that begin with PAYROLLPKG and are older than 30 days. The packages are deleted after they have been archived.

```
ARCHIVE PACKAGE PAYROLLPKG*
                TO DSNAME 'PAY.PACKAGE.ARCHIVE'
                OPTIONS WHERE OLDER THAN 30 DAYS
                        DELETE AFTER ARCHIVE.
```

The Backin Package Action

The BACKIN action reverses the BACKOUT action and returns outputs to a status of Executed. You can use the BACKIN action against a package that has a status of Executed and has been backed out.You can back in an entire package or specify an element action for backin.

Backin Package Syntax



The clauses BACKIN PACKAGE, STATEMENT NUMBER, and ELEMENT can be specified in any order. Name-masking cannot be used for any variable in the statement.

BACKIN PACKAGE ***package-id***

The BACKIN PACKAGE clause identifies the package you are backing-in. You must use a fully specified package ID. You can include imbedded spaces in the package ID. If the package ID contains an imbedded space or if the ID comprises only numeric characters (for example, 12345), then enclose the package ID in either single or double quotation marks.

STATEMENT NUMBER *nnnnn* ELEMENT *element-name*

Specifies the SCL statement number and the element name for the element action you want to back in. This clause limits the BACKIN PACKAGE clause by enabling you to specify the element actions you want to back in. If you want to back in all the element actions in the package, do not use this clause. The NUMBER keyword is optional. For each element action backed out, a separate BACKIN package action statement with PACkage, STATEment NUMber and ELEment clauses must be provided; only one package element action can be backed in per statement. Valid values for nnnnn are 1 – 65535. This batch syntax does not support long-named elements. You can backout or backin an individual long-named element in foreground.

Example: Backin Package SCL

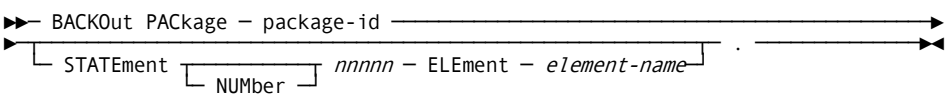
The following is an example of BACKIN PACKAGE SCL. The SCL backs in the package called PAYROLLPKG01.

```
BACKIN PACKAGE PAYROLLPKG01.
```

The Backout Package Action

The BACKOUT action allows a package to be backed-out after it has been executed. The BACKOUT action restores the executable and output modules to the status they were in prior to execution.You can back out an entire package or specify an element action for backout. You can use the BACKOUT action against a package only if the package has a status of Executed, In-execution, or Exec-failed and was created with the BACKOUT option enabled.

Backout Package Syntax



The clausesBACKOUT PACKAGE, STATEMENT NUMBER, and ELEMENT can be specified in any order. Name-masking cannot be used for any variable in the statement.

BACKOUT PACKAGE package-id

Specifies the package you want to back out. You must use a fully specified package ID. You can include imbedded spaces in the package ID. If the package ID contains an imbedded space or if the ID comprises only numeric characters (for example, 12345), then enclose the package ID in either single or double quotation marks.

STATEMENT NUMBER nnnnn ELEMENT element-name

Specifies the SCL statement number and the element name for the element action you want to back out. This clause limits the BACKOUT PACKAGE clause by enabling you to specify the element actions you want to back out. If you want to back out all the element actions in the package, do not use this clause. The NUMBER keyword is optional.For each element action you want to back out, a separate BACKOUT package action statement with PACkage, STATEment NUMber and ELEment clauses must be provided; only one package element action can be backed out per statement. Valid values for nnnnn are 1 – 65535. This batch syntax does not support long-named elements. You can backout or backin an individual long-named element in foreground.

Example: Backout Package SCL

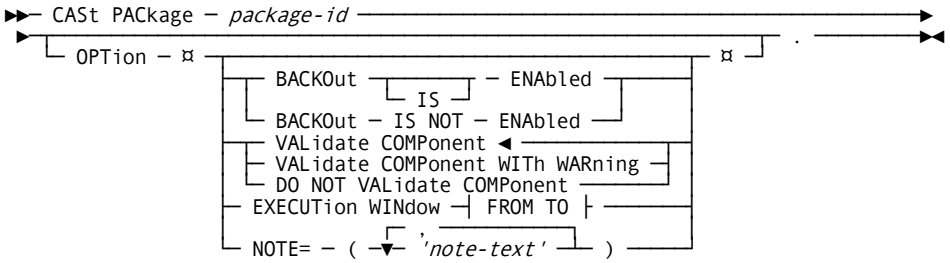
The following is an example of BACKOUT PACKAGE SCL. The SCL backs out the package called PAYROLLPKG01.

```
BACKOUT PACKAGE PAYROLLPKG01.
```

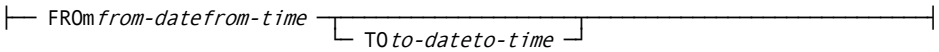
The Cast Package Action

The CAST action prepares the package for review and subsequent execution. Casting a package freezes the contents of the package and prevents further changes to the package. You can use the CAST action against a package that has a status of In-edit.

Cast Package Syntax



Expansion of FROM TO



CAST PACKAGE *package-id*

The CAST PACKAGE clause identifies the package you are casting. The package ID can be either fully specified, partially wildcarded or fully wildcarded.

You can include imbedded spaces in the package ID. If the package ID contains an imbedded space or if the ID is comprised of only numeric digits (for example, 12345) then enclose the package ID in either single or double quotation marks.

OPTIONS

Option clauses allow you to further specify package actions.

BACKOUT IS ENABLED/NOT ENABLED-The BACKOUT IS ENABLED/NOT ENABLED option indicates whether the backout facility will be available for this package.

VALIDATE COMPONENTS-The VALIDATE COMPONENTS option enables component validation within the package. You can only use this clause if your site allows you to specify whether component validation is to be performed.

The VALIDATE COMPONENTS clause causes the action to fail if component validation fails. The VALIDATE COMPONENTS WITH WARNING clause generates a warning if component validation fails. For information on component validation see the Packages Guide.

EXECUTION WINDOW FROM *from-date from-time* **TO** *to-date to-time*-The EXECUTION WINDOW clause allows you to change the execution window of the package as part of cast processing. You can use the EXECUTION WINDOW clause only if the package ID is fully qualified. Specify date values in DDMMMYY format and the time values in HH:MM format. If you specify the from-date, you must also specify the from-time. If you specify the to-date, you must also specify the to-time.

NOTES-Use the NOTES clause to add remarks to the package definition. Enclose the note text in either single or double quotation marks. If you use multiple text lines, enclose each text line in quotation marks and separate by commas. You can specify a maximum of 8 note text lines of up to 60 characters each. This text replaces any text that is already associated with the package.

Example: Cast Package SCL

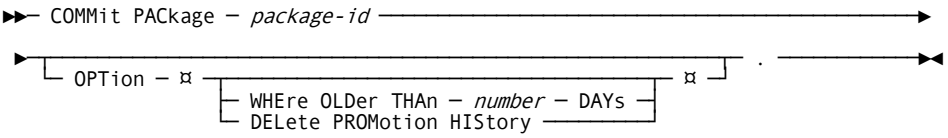
The following is an example of CAST PACKAGE SCL. The SCL casts a package called PAYROLLPKG01.

```
CAST PACKAGE PAYROLLPKG01.
```

The Commit Package Action

The COMMIT PACKAGE action removes all backout/backin data while retaining package event information. You can use the COMMIT action against a package only if the package has a status of Executed or Exec-failed.

Commit Package Syntax



COMMIT PACKAGE *package-id*

The COMMIT PACKAGE clause identifies the package you are committing. You can use a fully specified, partially wildcarded or fully wildcarded package ID. If you wildcard the package ID, you must specify the WHERE OLDER THAN clause. If you fully specify the package ID, the WHERE OLDER THAN clause is ignored.

You can include imbedded spaces in the package ID. If the package ID contains an imbedded space or comprises only numeric digits (for example, 12345), enclose the package ID in either single or double quotation marks.

OPTIONS

OPTION clauses allow you to further specify package actions.

WHERE OLDER THAN *number* **DAYS**-This clause allows you to specify the minimum age of the package you are committing. A package must be older than the number of days you specify in order to commit it. For example, if you specify WHERE OLDER THAN 30 DAYS and the current date is January 31, only packages executed successfully on or before January 1 are committed. There is no default value for the WHERE OLDER THAN clause. If you wildcard the package ID you must specify the WHERE OLDER THAN clause. The WHERE OLDER THAN value must be between 0 and 999, inclusive. You receive an error message if you specify a value outside this range.

DELETE PROMOTION HISTORY-This option is for promotion packages and it deletes all the promotion history associated with previous versions of the package.

Example: Commit Package SCL

The following are two examples of SCL for the COMMIT PACKAGE action. The first example commits a specific package called PAYROLLPKG01. The second example commits all packages that begin with PAYROLLPKG and are more than 30 days old.

Example One

```
COMMIT PACKAGE PAYROLLPKG01.
```

Example Two

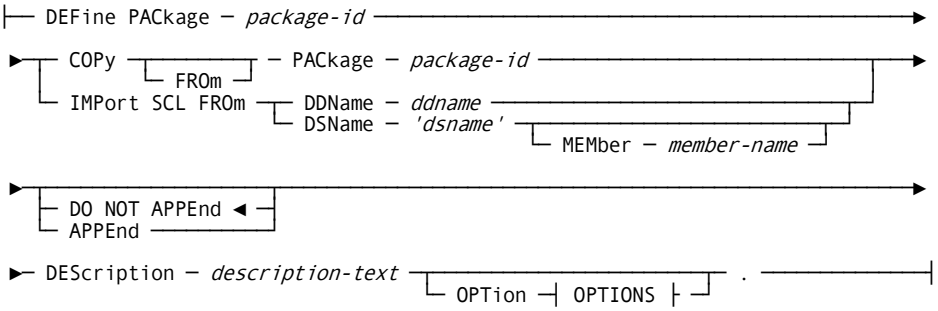
```
COMMIT  PACKAGE PAYROLLPKG*
                OPTIONS WHERE OLDER THAN 30 DAYS.
```

The Define Package Action

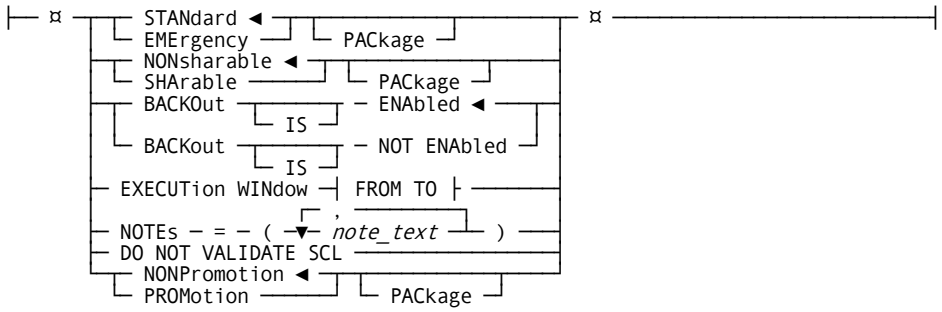
The DEFINE PACKAGE action creates a new package or updates an existing one. If you use the DEFINE PACKAGE action to update an existing package, the package must be in In-edit status.

Note: If you are using the DEFINE PACKAGE action to update an existing package and do not specify a DESCRIPTION, IMPORT SCL FROM, COPY PACKAGE or any OPTIONS, you will receive a caution-level message indicating that the update will not be performed because no information was provided to update the package.

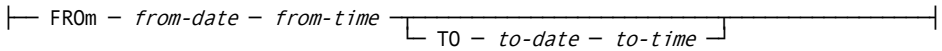
Define Package Syntax



Expansion of OPTIONS



Expansion of FROM TO



DEFINE PACKAGE *package-id*

The DEFINE PACKAGE clause identifies the package you are creating or updating. An update occurs if the package ID exists and a create occurs if it does not exist.

You must use a fully specified non-blank package ID. If you specify a blank package ID and have the GENPKGID exit defined, the GENPKGID exit invokes to generate a new package ID. If you do not have the GENPKGID exit installed or if the GENPKGID exit does not supply a package ID, an error message generates and the DEFINE PACKAGE action fails.

To specify a blank package ID place one or more blanks in single or double quotation marks as the DEFINE PACKAGE package-id statement. A blank package ID implies that the package is to be created.

Note: For more information about the GENPKID exit function, see the *Exits Guide*.

COPY FROM PACKAGE *package-id*

The COPY FROM PACKAGE clause directs the DEFINE action to copy the SCL from the package you specify into the package you are creating or updating. You must use a fully specified package ID.

If you are creating a new package you must specify either the COPY FROM PACKAGE or the IMPORT SCL FROM clause. If you are updating an existing package, the clauses are optional.

IMPORT SCL FROM

The IMPORT SCL FROM clause directs the DEFINE action to copy the SCL from the DD statement or data set name you specify into the package you are creating or updating.

If you are creating a new package you must specify either the COPY FROM PACKAGE or the IMPORT SCL FROM clause. If you are updating an existing package, the clauses are optional.

APPEND/DO NOT APPEND

The APPEND clause indicates whether to append the SCL you are adding to the existing package SCL or to replace it. You can only use the clause if you specify the COPY PACKAGE or IMPORT SCL FROM clauses. The default is DO NOT APPEND.

DESCRIPTION

The DESCRIPTION clause allows you to associate a 50-character description with the package. You must specify this clause if you are creating a new package. If you are updating an existing package the clause is optional. If the description text contains imbedded spaces enclose it in single quotation marks. The description text you enter is not converted to uppercase. Lowercase characters remain in lowercase.

OPTIONS

OPTION clauses allow you to further specify package actions.

STANDARD/EMERGENCY PACKAGE-This option allows you to specify the package type. If you do not specify the STANDARD/EMERGENCY PACKAGE clause and you are creating a new package, the package defaults to a STANDARD package.

SHARABLE/NONSHARABLE PACKAGE-This option allows you to specify whether this package can be edited by more than one person when in In-edit status. If the package is sharable it can be edited by someone other than the package creator. If the package is non-sharable it can only be edited by its creator. If you do not specify the SHARABLE/NONSHARABLE PACKAGE clause and you are creating a new package, the package defaults to a NONSHARABLE package.

BACKOUT IS ENABLED/NOT ENABLED-The BACKOUT IS ENABLED/NOT ENABLED option indicates whether you wish to have the backout facility available for this package. Use this clause when creating a new package only. The default is BACKOUT IS ENABLED.

PROMotion/NONPromotion PACkage-This option lets you define the package as a promotion package or a nonpromotion package. If you do not specify the PROMOTION/NONPROMOTION PACKAGE clause and you are creating a new package, the package defaults to a NONPROMOTION package.

EXECUTION WINDOW FROM *from-date from-time* **TO** *to-date to-time*-This option allows you to specify the time frame within which to execute the package. Specify date values in DDMMMYY format and the time values in HH:MM format.

If you specify the from-date, you must also specify the from-time. If you specify neither the from-date nor the from-time and you are creating a new package, the from-date and the from-time default to the current date and time, respectively.

If you specify the to-date, you must also specify the to-time. If you specify neither the to-date nor the to-time and you are creating a new package, the to-date and the to-time default to 31DEC79 and 00:00, respectively.

NOTES-Use the NOTES clause to add remarks to the package definition. Enclose the note text in either single or double quotation marks. If you use multiple text lines, enclose all the lines in parentheses () and enclose each text line in single quotation marks, separated with a comma. You can specify up to 8 text lines of up to 60 characters each. A note line cannot be split across syntax lines (that is, the beginning and ending quote must be on the same line). The text including the last quote cannot exceed column 72. The text replaces any text which is already associated with the package.

DO NOT VALIDATE SCL-If you specify the DO NOT VALIDATE SCL option, the package components are not validated while creating or updating a package.

Example: Define Package SCL

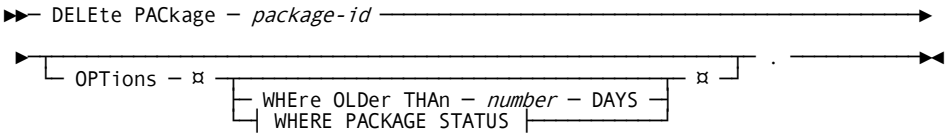
The following is an example of DEFINE PACKAGE SCL. The SCL defines a new package called PAYROLLPKG01, to be used to implement a new payroll system. The SCL is copied from the data set specified by the IMPORT SCL FROM DSNAME clause.

```
DEFINE  PACKAGE PAYROLLPKG01
                DESCRIPTION 'PACKAGE TO IMPLEMENT THE NEW PAYROLL SYSTEM'
                IMPORT SCL FROM DSNAME 'PAY.PACKAGE.SCL' MEMBER 'ADDSCL01'
                OPTIONS EXECUTION WINDOW FROM 01JAN93 00:01 TO 31DEC93 23:59
                BACKOUT IS ENABLED
                SHARABLE PACKAGE
                STANDARD PACKAGE
                NOTES=('THIS PACKAGE IMPLEMENTS THE NEW PAYROLL SYSTEM.',
                'THE SCL FOR THIS PACKAGE WAS IMPORTED','FROM SERVICE PACK SP01.',
                'THE PACKAGE MUST BE CAST AND APPROVED BY JUNE 30.').
```

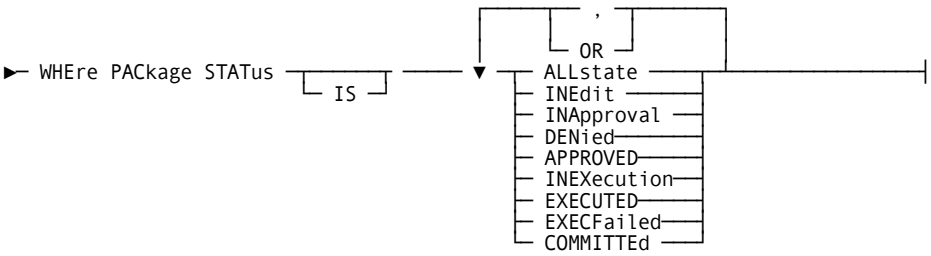
The Delete Package Action

The DELETE PACKAGE action allows you to delete packages. You can use the DELETE PACKAGE action to delete packages of any status type.

Delete Package Syntax



Expansion of WHERE PACKAGE STATUS



DELETE PACKAGE *package-id*

The DELETE PACKAGE clause identifies the package you are deleting. You can use a fully specified, partially wildcarded or fully wildcarded package ID. If you wildcard the package ID, you must specify the WHERE PACKAGE STATUS IS clause. If you use a fully specified package ID, the WHERE PACKAGE STATUS IS and the WHERE OLDER THAN clauses are ignored.

OPTIONS

OPTION clauses allow you to further specify package actions.

WHERE OLDER THAN *number* **DAYS**-Use this clause to specify the minimum age of the packages you are deleting. A package must be older than the number of days you specify in order to delete it. This number of days is compared against the create date of the package to determine if the package meets the criterion of the WHERE OLDER THAN clause.

WHERE PACKAGE STATUS-This clause specifies the statuses of the packages you are deleting. You can only use this clause when you wildcard the package ID. If you do not specify the WHERE PACKAGE STATUS clause, the DELETE PACKAGE action deletes packages of any status type.

Example: Delete Package SCL

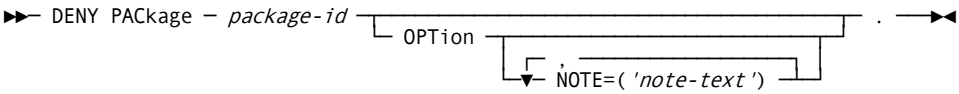
The following is an example of DELETE PACKAGE SCL. The SCL deletes all packages that begin with PAYROLLPKG, are older than 30 days, and are in the In-edit status.

```
DELETE  PACKAGE PAYROLLPKG*
                OPTIONS WHERE OLDER THAN 30 DAYS
        WHERE PACKAGE STATUS IS INEDIT.
```

The Deny Package Action

The DENY PACKAGE action changes the status of a package to Denied. You can use the DENY action against a package that has a status of In-approval.

Deny Package Syntax



DENY PACKAGE *package-id*

The DENY PACKAGE clause identifies the package you wish to deny. You must use a fully specified package ID.

OPTIONS

OPTION clauses allow you to further specify package actions.

NOTES-You can use the NOTES clause to add remarks to the package definition. Enclose the note text in either single or double quotation marks. If you use multiple text lines, enclose each text line in quotation marks, and separate by commas. You can specify up to 8 text lines of up to 60 characters each. The text replaces any text that is already associated with the package.

Example: Deny Package SCL

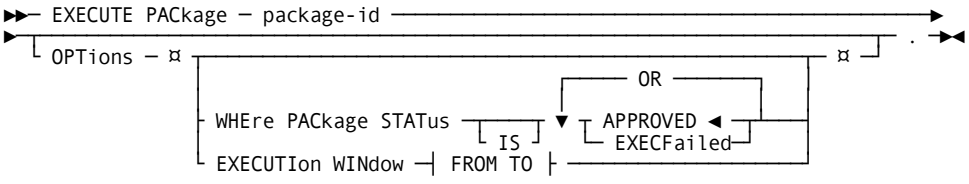
The following is an example of DENY PACKAGE SCL. The SCL denies the package called PAYROLLPKG01 and replaces any package notes associated with the package.

```
DENY PACKAGE PAYROLLPKG01
        OPTIONS NOTES=('THE PACKAGE WAS DENIED BECAUSE ALL OF THE DOC- ',
'UMENTATION WAS NOT INCLUDED IN THE PACKAGE.',
'NOTE: THESE NOTES WILL REPLACE ANY EXISTING',
'PACKAGE NOTES.').
```

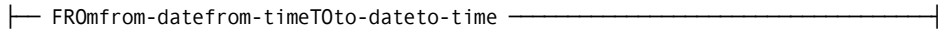
The Execute Package Action

The EXECUTE PACKAGE action executes a package. You can use the EXECUTE PACKAGE action against packages that have a status of Approved or Execfailed. The default is to only execute approved packages.

Execute Package Syntax



Expansion of FROM TO



EXECUTE PACKAGE *package-id*

The EXECUTE PACKAGE clause identifies the package you want to execute. You can use a fully specified, partially wildcarded, or fully wildcarded package ID. When the package-id is fully specified, the WHERE PACKAGE STATUS clause will be ignored and the parser will issue a warning message.

OPTIONS

OPTION clauses allow you to further specify action requests.

EXECUTION WINDOW FROM *from-date from-time* **TO** *to-date to-time*

Specifies the time frame within which to execute the package. Specify date values in DDMMMYY format and the time values in HH:MM format. If you specify the from-date, you must also specify the from-time. If you specify the to-date, you must also specify the to-time. You can only use the EXECUTION WINDOW clause if the package is fully qualified and the existing execution window is closed.

WHERE PACKAGE STATUS

Specifies the statuses of the packages you want to execute. You can only use this clause when you wildcard the package-id. The default is to execute packages that have a status of Approved. Use the WHERE PACKAGE STATUS IS EXECFAILED clause to re-execute packages that have previously failed. Use the OR clause to indicate that packages of either status should be executed.

Example: Execute Package SCL

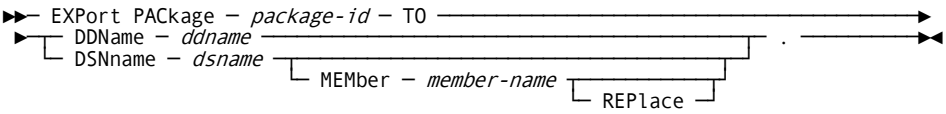
The following is an example of EXECUTE PACKAGE SCL. The SCL executes the package called PAYROLLPKG01.

```
EXECUTE PACKAGE PAYROLLPKG01.
```

The Export Package Action

The EXPORT PACKAGE action writes the SCL associated with a package to an external data set. You can use the EXPORT PACKAGE action against a package of any status type.

Export Package Syntax



EXPORT PACKAGE *package-id*

The EXPORT PACKAGE clause identifies the package you are exporting. You must use a fully specified package ID.

TO DDNAME ddname
DSNAME dsname
MEMBER member-name
REPLACE

The TO clause identifies where to write the package SCL. Enter either a DDname or a data set name, not both. The data set defined by the TO DDNAME/DSNAME clause must be allocated with either fixed or variable length records. If fixed, the record length must be 80. If variable, the record length must be at least 84.

The TO DDNAME clause identifies the name of an allocated DD statement. The DD statement must reference a sequential data set or a partitioned data set with an explicit member.

The TO DSNAME clause identifies the name of an existing catalogued data set. If the data set is a partitioned data set, you can use the member clause to specify a member name to be created. If you do no specify a MEMBER clause, the member name created is TEMPNAME. The REPLACE clause replaces an existing like-named member. You can only use the REPLACE clause if the MEMBER clause is specified.

Example: Export Package SCL

The following is an example of EXPORT PACKAGE SCL. The SCL exports the package SCL to the data set called PAY.PACKAGE.SCL. The MEMBER clause specifies a member to be created called PAYPKGO1. It replaces any like-named member.

```
EXPORT PACKAGE PAYROLLPKG01
   TO DSNAME 'PAY.PACKAGE.SCL' MEMBER 'PAYPKG01' REPLACE.
```

The Inspect Package Action

The Inspect Package action checks the elements contained within a package for conflicts that would effect its successful execution. The Inspect action checks for security, signout, and synchronization conflicts as well as changes in source. You can use the Inspect Package action to inspect packages that have the following status:

- In-approval
- Approved
- Exec-failed

Inspect Package Syntax

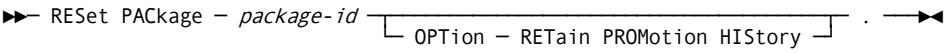
INSPECT PACKAGE *package-id*

The INSPECT PACKAGE clause identifies the package you are inspecting. You can use a fully specified, partially wildcarded, or fully wildcarded package ID. If you partially or fully wildcard the package ID, CA Endevor SCM only inspects packages that have a status of in-approval, approved, or execution failed.

The Reset Package Action

The RESET PACKAGE action allows you to set the status of a package back to In-edit so you can modify it.You can use the RESET action against a package of any status type.

Reset Package Syntax



RESET PACKAGE *package-id*

The RESET PACKAGE clause identifies the package you are resetting. You must use a fully specified package ID. The RESET PACKAGE action resets a package of any status type.

Example: Reset Package SCL

The following is an example of RESET PACKAGE SCL. The SCL resets the package called PAYROLLPKGO1 back to the status of In-edit.

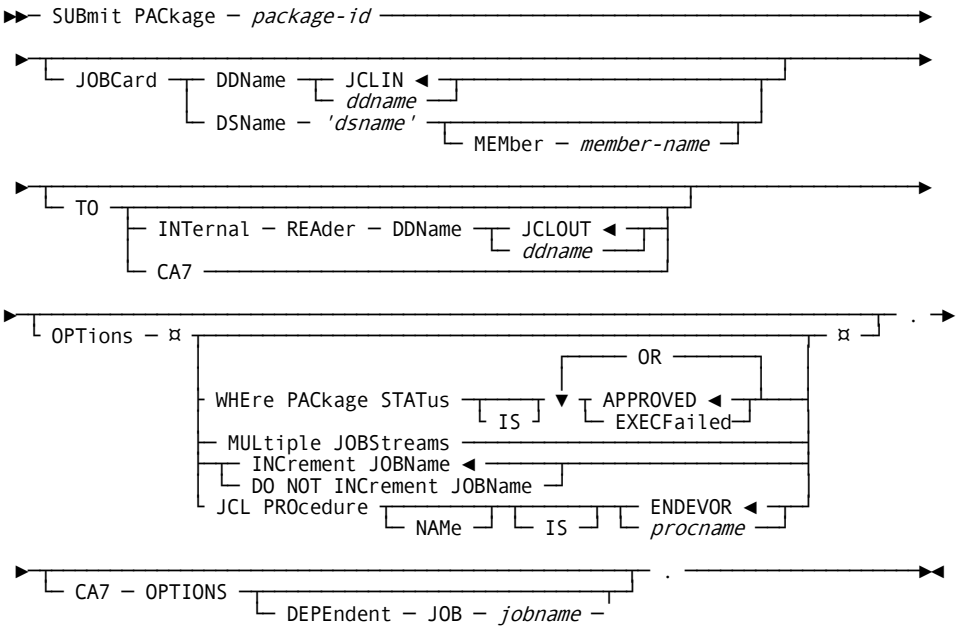
```
RESET PACKAGE PAYROLLPKG01.
```

The Submit Package Action

Use the SUBMIT PACKAGE action to submit a JCL job stream to execute one or more packages. The SUBMIT PACKAGE action is a replacement for the existing Batch Package Submission Utility C1BM6000. Users of C1BM6000 *must* migrate from C1BM6000 to ENBP1000.

Note: For more information about ENBP1000 and the SUBMIT PACKAGE action, see the *Packages Guide*.

Submit Package Syntax



SUBMIT PACKAGE *package-id*

The SUBMIT PACKAGE clause identifies the package you are submitting for execution. You can use a fully specified, partially wildcarded, or fully wildcarded package ID. When the package-id is fully specified, the WHERE PACKAGE STATUS clause will be ignored and the parser will issue a warning message.

```
JOBCARD DDNAME JCLIN

        DDNAME ddname

         DSNAME dsname

           MEMBER member-name
```

The JOBCARD clause identifies the location of the data set that contains the JCL jobcard. The location be either a DDname or data set name. If you do not specify the JOBCARD clause, the JCLIN DD statement is used by default as the JCL jobcard location. If you specify this clause, the clause must identify either a sequential or a partitioned data set with an explicitly specified member name. The data set must have fixed length records and the record length (LRECL) must be exactly 80.

INTERNAL READER DDNAME

The INTERNAL READER DDNAME clause identifies the name of a pre-allocated DD statement to which the JCL is written. Generally, this is allocated as DD SYSOUT=(class,INTRDR). The DD statement can also reference any sequential or partitioned data set with an explicit member that has a record length (LRECL) of 80. If you do not specify the INTERNAL READER DDNAME clause, the SUBMIT PACKAGE action writes the JCL to the JCLOUT DD statement.

CA7

If the TO CA7 clause is specified, neither the MULTIPLE JOBSTREAMS nor the INCREMENT JOBNAME parameters may be specified.

If the TO CA7 clause is not specified, any specifications for CA7 options will be ignored.

OPTIONS

OPTION clauses allow you to further specify package actions.

WHERE PACKAGE STATUS -Specifies the statuses of the packages you want to submit. You can only use this clause when you wildcard the package-id. The default is to submit packages that have a status of Approved. Use the WHERE PACKAGE STATUS IS EXECFAILED clause to re-execute packages that have previously failed. Use the OR clause to indicate that packages of either status should be submitted.

MULTIPLE JOBSTREAMS-Use the MULTIPLE JOBSTREAMS clause to submit a separate, unique job for each package. This clause is equivalent to the C1BM6000 MULTJOBS JCL parameter. If you do not specify this clause a single job that has a unique job step for each package is submitted. A maximum of 200 packages can be processed in a single job stream. If more than 200 packages meet the selection criteria, the SUBMIT action submits additional jobs, each with up to 200 steps, until all of the eligible packages have been submitted.

Note: C1BM6000 has been replaced by the Batch Facility (ENBP1000). Users of C1BM6000 *must* migrate from C1BM6000 to ENBP1000.

INCREMENT JOBNAME-The INCREMENT JOBNAME clause directs the Batch Package Facility to increment the last character in the jobcard you provide. In general, use the INCREMENT JOBNAME clause with the MULTIPLE JOBSTREAM clause to increment the last character in the JCL jobcard for each job stream you submit.

You can also use the INCREMENT JOBNAME clause when you submit a single job stream and more than 200 eligible packages are found. If an additional job stream is created, the INCREMENT JOBNAME clause controls whether the additional job names are incremented. The SUBMIT action uses the following rules when incrementing the last character in the job name:

- If the character is numeric, the next number is selected with wrap-around to '0'.
- If the character is alphabetic, the next letter is selected with wrap-around to 'A'.
- If the character is neither numeric nor alphabetic, it is not incremented.

JCL PROCEDURE NAME-The JCL PROCEDURE NAME clause identifies the name of the JCL procedure you wish to invoke in the SUBMIT PACKAGE action JCL. The name must be one to eight characters long and constructed to accept the package ID as the only parm. The symbolic to pass the package ID is PKGID. For example:

```
//ENDEVOR PROC  PKGID=
//PS01  EXEC  PGM=NDVRC1,PARM=(C1BM3000,,&PKGID).
```

If you do not specify the JCL procedure name the SUBMIT PACKAGE action creates JCL to invoke the ENDEVOR procedure. A sample JCL procedure is supplied in iprfx.iqual.CSIQJCL member ENDEVOR. Copy this procedure into an existing PROCLIB defined to JES on your system, or use it as a model for your own JCL batch package PROC.

CA7 OPTIONS DEPENDENT JOB jobname

DEPENDENT JOB Specifies a single predecessor job which must complete while demanded job is waiting.

For example: CA7 OPTIONS DEPENDENT JOB CICSDOWN-This means that package ENDEVOR will not run until jobname cicsdown is successfully completed.

```
SCL Example:
     SUBMIT PACKAGE 'ENDEVOR'
      JOBCARD DSNAME 'BST.ENDEVOR.JCLLIB'
          MEMBER 'JOBCARD'
          TO CA7
          OPTIONS WHERE PACKAGE STATUS IS APPROVED
         JCL PROCEDURE NAME IS ENDEVOR
          CA7 OPTIONS DEPENDENT JOB CICSDOWN.
```

CA7 translates the above SCL into CA7 syntax similar to this:

```
   DEMAND,JOB=PILRO01C,DEPJOB=CICSDOWN,
        SET=NDB,JCLLIB=&ENDEVOR.
```

Note: For more information about CA 7 DEMAND, see the CA7 commands manual or contact you local CA7 Support.

Example: Submit Package SCL

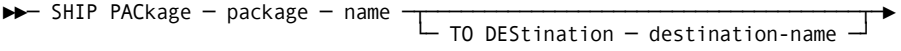
The following is an example of SUBMIT PACKAGE SCL. The SCL submits for execution all packages that begin with PAYROLLPKG. Each package will have a unique jobstep, the jobname will not be incremented.

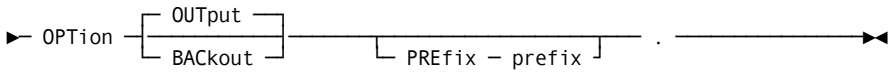
```
SUBMIT PACKAGE PAYROLLPKG*
        OPTIONS DO NOT INCREMENT JOBNAME.
```

The Ship Package Action

Use the Ship Package statement to ship a package to a remote site. You can ship the package output members or the package backout members. The Ship Package statement lets you indicate the package you want to ship, the destination you want to ship to, whether you want to ship output or backout members, and the data set name prefix to be used in the XCOM or CONNECT:DIRECT transmission methods.

This statement has the following syntax:





SHIP PACKAGE *package-name*

Indicates the one- to 16-character name of the package you want to ship.

TO DESTINATION *destination-name*

Indicates the one- to eight-character name of the remote site to which you want to ship the specified package. The destination name can be alphanumeric, but must begin with an alphabetic character.

This information is required. If you do not enter a destination here, a SET DESTINATION statement with the appropriate information must have been previously coded.

OPTION OUTPUT/BACKOUT

Indicates whether you want to ship *output* members or *backout* members to the remote site. If you do not indicate an option here, the system looks for a SET OPTION statement. If no SET OPTION clause is found, the system defaults to Option Output, and automatically ships output members to the remote site.

OUTPUT

Indicates that you want to ship the members created by the execution of the package.

BACKOUT

Indicates that you want to ship the members needed to backout a package.

PREfix *prefix*

Indicates the one- to eight-character data set name prefix to be used in the XCOM or CONNECT:DIRECT transmission methods. If the transmission method selected is not XCOM or CONNECT:DIRECT, this option is ignored. More than one node can be specified for the Package Ship output prefix, as long as the nodes, with separating periods, fit within the eight-character space.

Chapter 6: Managing Environments in Batch

This section contains the following topics:

How to Manage Environments in Batch

After your lifecycle Environments are defined in the Defaults table (C1DEFLTS) you must define objects to each Environment. These objects are as follows:

- The inventory structure objects, which include Systems, Subsystems, and Types.
- Other objects that are associated to an Environment. These objects can include Approver groups, Approver group relations, Processor groups, Processor symbols, and a Type Sequence.

If you choose to enable the Package Ship facility, you will need to define the Package Ship objects: Package Shipment Destinations, Package Shipment Map Rules, and Package Shipment Map Rules for USS supported files. However, the Package Ship objects are not defined to a particular Environment.

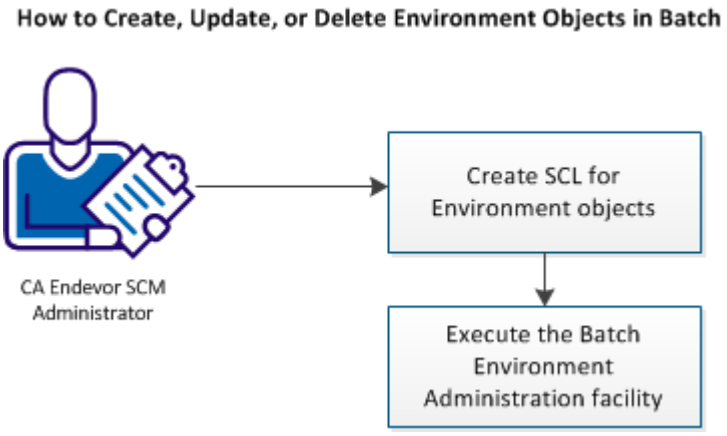
As an administrator, you can manage the Environment objects and Package Ship objects in batch processing mode using the Batch Environment Administration facility (Batch Admin). Most of the objects can be created, updated, or deleted in batch. Alternatively, you can manage the objects using the foreground panels. However, Batch Admin is more efficient. Batch Admin eliminates the need to navigate multiple screens, lets you build new objects based on existing ones, and supports name-masking on object values which automatically expands a single statement into multiple actions.

In addition to managing Environment objects, you can use Batch Admin to create a new Environment based on an existing Environment, including its inventory structure and associated objects.

Batch Admin enables you to manage Environment objects in batch by executing the following types of SCL statements:

- **Build statements**— Create Define statements from an *existing* object and its subordinates.
- **Define statements**— Create a new, or update an existing, object. You can create Define statements for a single Environment object (for example, a System, Subsystem, or Type) or for an entire Environment structure (for example, all the objects for the Production Environment).
- **Delete statements**— Delete an existing object.

You can execute multiple Define, Build, and Delete statements in the same Batch Admin job or even the same job step. However, if you execute a Build statement, which creates a Define statement, you will need to execute Batch Admin again to execute the Define statement. For example, if you execute a Build statement for a subsystem in order to define a new subsystem, you would: 1) execute the build in one job; 2) edit the Define statement to change the subsystem object name to the new name; and then 3) execute Batch Admin again to execute the modified Define statement.



Complete these steps:

1. Create SCL for Environment definitions (see page 244).
2. Execute the Batch Environment Administration facility (see page 245).

Example: Create a New System

Suppose you want to start a new release of a software application. You need a new inventory location for the new code. Since you already have a System defined for the prior release, you can easily create a new System from the old one. The following example shows how to do this:

1. Code the Build SCL for System statement.

   The following is an example of the Build SCL for System statement. The example builds Define SCL for the System ACCT using Environment DEVEL. The optional clause Include Subordinates is specified. Therefore, Define SCL is built for all inventory definitions associated with System ACCT. The SCL is written to the PDS data set named ENDEVOR.SCLOUT in member SYSACCT. The Replace clause will replace the contents of the member if the member already exists in the PDS. If you do *not* use this clause and the member already exists in the PDS, you will get an error.

```
BUILD SCL FOR SYSTEM "ACCT"
    FROM ENVIRONMENT "DEVEL"
    INCLUDE SUBORDINATES
    TO DSNAME "ENDEVOR.SCLOUT"
        MEMBER "SYSACCT"
                    REPLACE .
```

2. Update the ENESCLIN DD statement in the Batch Admin job. Sample JCL is located in the *iprfx.iqual.*CSIQJCL installation library, member name ENBE1000. Add the statement instream or reference the data set where the Build SCL for System statement is located. Execute the job.

   Define SCL statements for the ACCT System along with Define statements for all the ACCT subordinates (Subsystems, Types, Processor Groups, and so on) are written to ENDEVOR.SCLOUT(SYSACCT).

3. Edit the Define SCL statements that were built and perform a global change of the old System name to the new System name for the new release. Update the ENESCLIN DD statement in Batch Admin JCL ENBE1000 to reference the location of the Define SCL statements, ENDEVOR.SCLOUT(SYSACCT). Execute the job.

   The new Environment objects are now defined to CA Endevor SCM. Other considerations such as data set naming conventions and data set allocations must be addressed.

Create SCL for Environment Objects

To manage Environment objects and Package Ship objects in batch, you must code SCL statements to control the actions you want to perform.

Code the Build, Define, or Delete statement syntax for the object that you want to create, update, or delete. The Batch Admin job can refer to the SCL statements as instream data, a sequential data set, or a partitioned data set. To create the SCL, you can use the following methods:

- Write the SCL statements following the syntax rules. For more information, see Build Statements (see page 250), Define Statements (see page 266), or Delete Statements (see page 305).
- Use the ISPF/PDF edit macros to create model Build, Define, and Delete statements defined with default values. Then, edit the default values as appropriate. For more information about the edit macros, see Batch Admin Edit Macros (see page 247).

Code the SCL as appropriate to perform the task you want to perform. The following tasks can be performed for the Environment objects: Approver Groups, Approver Relations, Systems, Subsystems, Types, Processor Groups, Processor Symbols, Environments, and Type Sequence:

- *To create a new Environment object based on an existing object definition,* code the Build statement for the existing definition. After the job has run and created the Define SCL, edit the Define statements to specify the name of the new definitions. Also, update any data set names as appropriate for the new definitions.
- *To update an existing Environment object definition,* code the Build statement for the definition you want to update. After the job has run and created the Define SCL, edit the Define statements to specify the change you want to make.

- *To create a new Environment object*, code the appropriate Define statement.
- *To delete an existing Environment object*, code the appropriate Delete statement.

Execute the Batch Environment Administration Facility

The Batch Environment Administration facility (Batch Admin), sample JCL member ENBE1000, lets you administer Environment objects and Package Ship objects in batch mode by executing SCL statements specified in the ENESCLIN DD statement.

The following general rules apply to ENBE1000 execution:

- There is no defined limit to the number of statements you can specify and process in a single execution.
- Statements are executed in the sequence you provide.
- Statements are parsed before execution.
- If any syntax errors are found, none of the statements are processed.
- Statements are processed as long as the action return code is less than or equal to 12. If a return of greater than 12 is received all remaining statements are bypassed.

Follow these steps:

1. Locate the Batch Environment Administration facility. Sample JCL is located in the *iprfx.iqual*.CSIQJCL member name ENBE1000.
2. Edit the following DD statements as needed:

   ENESCLIN

   Defines the Batch Environment Administration facility SCL statements. The DD statement can refer to instream data, a sequential data set or a partitioned data set with an explicit member.

   If the ENESCLIN DD statement refers to a data set, the data set must have either fixed length or variable length records. If the records are fixed length, the record length must be exactly 80. If the records are variable length, the record length must be at least 84. If any of the data set attributes are incorrect, an error message is written and a return code of 12 is set.

   C1MSGS1

   Defines the destination of the Batch Environment Administration facility execution reports. By default the Action Execution report and the Action Summary report are written to the C1MSGS1 DD statement. If you code the C1MSGS1 and C1MSGS2 DD statements the Action Execution Report is written to the C1MSGS1 file and the Execution Summary report is written to the C1MSGS2 file.

3. (Optional) Code the validate mode instead of execute mode. To do this, change the PARM= statement on the sample JCL to PARM='ENBE1000VALIDATE'.

   Validate mode enables you check the syntax of your SCL statements before submitting them for execution. In validate mode, the statements specified in the ENESCLIN DD statement are parsed, but not executed.

4. Add a job card and execute the ENBE1000 job.

5. See the return code and execution reports. The job can result in the following return codes and execution reports:

   - The Batch Environment Administration facility passes one of the following return codes after execution is complete:

     0— All actions were performed successfully.

     4— One or more actions completed with a warning message.

     8— One or more actions completed with a caution message.

     12— One or more actions completed with an error message. The action may not have completed successfully.

     16— An unrecoverable error occurred.

     20— The C1MSGS1 DD statement was not allocated or the C1MSGS1 file could not be initialized.

   - As the Batch Environment Administration facility is processing, CA Endevor SCM writes a report to the C1MSGS1 DD statement. The report is divided into the following sections:

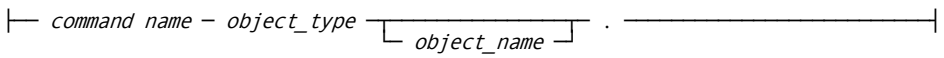
     The Statement Summary Report— When you submit batch Environment definition actions, CA Endevor SCM validates the SCL syntax and assigns a statement number to each SCL statement. The Statement Summary Report lists your SCL statements and error messages. If no errors are detected, processing continues and the Action Execution Report and the Action Summary Report are produced. If any errors do exist, processing terminates.

     The Action Execution Report— Lists the messages generated by each action during its processing.

     The Action Summary Report— Summarizes the actions performed by the Batch Environment Administration facility. The report contains one line for each Environment object processed by each action. The report line identifies the action, the Environment object, and the action return code. The Stmt Number refers to the number this statement is assigned in the Action Summary Report. The Action Number refers to the action number assigned to this request in the Execution Report. If specified, this report is written to the C1MSGS2 data set.

Batch Admin Edit Macros

Use the Batch Environment Administration facility (Batch Admin) edit commands to create model SCL statements for the Build, Define, and Delete SCL statements. These commands are implemented as ISPF/PDF edit macros.You can invoke the commands in an ISPF/PDF edit session that has the CA Endevor SCM *iprfx.iqual*.CSIQCLS0 installation library allocated to its SYSPROC concatenation. The edit macros are written in REXX. Therefore, they are only available on TSO/E Version 2 or higher and ISPF/PDF Version 2.3 or higher. To invoke a command, open a blank member in a PDS dataset and execute the command name and object type. The syntax for the edit command is:



Command Name

Identifies the SCL action you are creating and places model SCL statements at the current cursor location of the data set you are editing. The following general rules apply to the SCL created by the edit commands:

- SCL actions that contain optional clauses with default values are generated using the default value. For example, the Define System action includes the optional clause Comments Not Required.
- SCL actions that contain required clauses are generated with a place holder in which you can enter the appropriate information. The place holder is a string of question marks ('?'). For example, the ENBUILD command generates the following To Dsname clause: TO DSNAME ????????? .
- There is one SCL clause on each data record. If necessary, an SCL clause can span multiple lines.
- The ENDEFINE command and the ENBUILD command generate the Stage Number, instead of the Stage ID, on the To clause.

Specify one of the following commands:

ENDEFINE— Generates model SCL statements for DEFINE statements.

ENDELETE— Generates model SCL statements for DELETE statements.

ENBUILD— Generates model SCL statements for BUILD statements.

Note: For more information about the edit macros, enter ENBUILD HELP, ENDEFINE HELP or ENDELETE HELP on the ISPF/PDF edit session command line to view tutorials.

Object_Type

Identifies the kind of environment definition that you are creating.

The following table provides a list of valid object types and their abbreviations and the corresponding environment definition created. Object types are only valid for the commands marked with an "x" in the table below.

Object_type	Object_type Abbreviations	Environment Definition Created	Valid for ENDEFINE	Valid for ENDELETE	Valid for ENBUILD
SYSTEM	SYS	System	x	x	x
SUBSYSTEM	SUB	Subsystem	x	x	x
TYPE	TYP	Element Type	x	x	x
APPROVER	APP	Approver Group	x	x	x
RELATION	REL	Approver Group Relation	x	x	x
GROUP	GRO	Processor Group	x	x	x
SYMBOL	SYM	Processor Symbol	x	x	x
SEQUENCE	SEQ	Type Sequence*	x		x
ENVIRONMENT	ENV	Environment*			x
DESTINATION	DES	Package Shipment Destination	x	x	x
MAPRULE	MAP	Package Shipment Map Rule for Data Sets*	x	x	
USSMAPRULE	USS	Package Shipment Map Rule for USS Path Names*	x	x	

***Note:**

Type Sequence: You cannot create or delete a Type Sequence. You can only update and build based on an exisiting Type Sequence.

Environment: You cannot delete an Environment. You can use ENBUILD to generate Define SCL that can be used to update the subordinate objects of the Environment or to define a new Environment based on the existing Environment and its subordinate objects.

Package Shipment Map Rules: You can only define or delete map rules. You cannot build a map rule based on an existing map rule.

Object_Name

(Optional.) Names the environment definition for which you are creating SCL. The macro substitutes the object_name into the SCL generated.

Examples: Batch Admin Edit Macro Commands

The following examples show how to code Batch Admin edit macro commands in an ISPF/PDF edit session to create SCL statements.

- To create Define SCL for a System, enter one of the following:
  - ENDEFINE SYSTEM
  - ENDEFINE SYS

  The result is a model Define System statement coded with default options. All variable values are indicated by question marks to indicate where you must specify a value. You can change the default options.

```
DEFINE SYSTEM '????????'
  TO ENVIRONMENT '????????'
  DESCRIPTION '???????? '
  NEXT SYSTEM '????????'
  COMMENTS NOT REQUIRED
  CCID NOT REQUIRED
  DUPLICATE ELEMENT CHECK IS NOT ACTIVE
  DUPLICATE PROCESSOR OUTPUT CHECK IS NOT ACTIVE
  ELEMENT JUMP ACKNOWLEDGMENT REQUIRED
  SIGNOUT IS NOT ACTIVE
  SIGNOUT DATASET VALIDATION IS NOT ACTIVE
  STAGE ONE LOAD LIBRARY IS
     '???????? '
  STAGE ONE LIST LIBRARY IS
     '???????? '
  STAGE TWO LOAD LIBRARY IS
     '???????? '
  STAGE TWO LIST LIBRARY IS
     '???????? '
```

- To create a Delete statement for a Subsystem named SUB1, enter one of the following:
  - ENDELETE SUBSYSTEM SUB1
  - ENDELETE SUB SUB1

  The result is a model Delete Subsystem statement coded with the Subsystem name you specified. The Environment and System names need to be added. Also, you can edit the object name SUB1, if you decided on another name.

```
DELETE SUBSYSTEM 'SUB1'
  FROM ENVIRONMENT '????????'
       SYSTEM '????????'
```

Build Statements

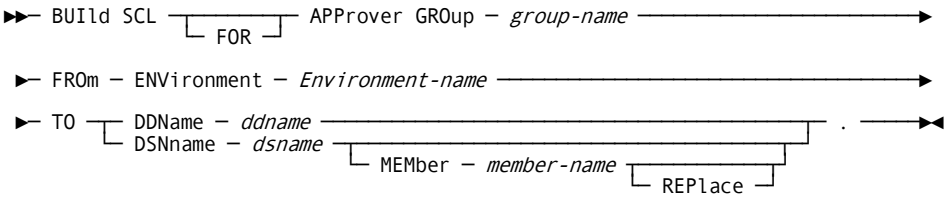
Use BUILD statements to create DEFINE statements from an existing Environment. You can create DEFINE statements for a single Environment definition (for example a System, Subsystem, or Type) or for an entire Environment structure (for example the Production Environment). The statements are written to a sequential data set or a partitioned data set member. You can modify the data set that the BUILD action creates.

The following general conventions apply to all BUILD statements:

- The Environment names you specify in the BUILD statement can have a maximum of eight characters with the exception of SHIPMENT DESTINATION, which can be no longer than seven characters, and APPROVER GROUP, which can be no longer than 16 characters.
- The Environment names you specify in the BUILD statement cannot include embedded spaces and must consist of national characters (A-Z, 0-9, @, # or $).
- Name-masking is supported on some variable values. For details, see the product documentation for each Build statement.
- If the data set name you specify in the TO DSNAME clause contains embedded periods you must enclose it in quotation marks.
- Variable values can be enclosed in double or single quotes, but quotes are not required, except for data set names. All variable values on Define statements created by Build statements are enclosed in quotes by default.

Build SCL for Approver Group Syntax

Use the Build SCL for Approver Group statement to build DEFINE statements for specified approver groups. This SCL statement has the following syntax:



Note: You can use name-masking to specify *some* variable values. A name-masked value is not explicit, because it includes a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. For more information about name-masking, see Name-Masking (see page 23).

BUILD SCL FOR APPROVER GROUP *group-name*

Builds DEFINE statements for the specified approver groups. The FOR keyword is optional. You can specify a name-masked approver group name.

FROM ENVIRONMENT *Environment-name*

Identifies the Environment location of the approver group.You can specify a name-masked Environment name.

TO DDNAME *ddname* | DSNAME *dsname* [MEMBER *member-name* REPLACE]

Indicates where you want the BUILD SCL written to. The data set defined by this clause must be allocated with either fixed or variable length records. If fixed, the record length must be 80. If variable, the record length must be at least 84. Specify either the DDNAME parameter or the DSNAME parameter, but not both.

DDNAME ddname

Identifies the name of an allocated DD statement. The DD statement must define a sequential data set or a partitioned data set with an explicit member.

DSNAME *dsname* [MEMBER *member-name* REPLACE]

Identifies the name of an existing catalogued data set. If the data set name contains imbedded periods, enclose it in quotation marks. If the data set is a partitioned data set, you can use the MEMBER clause to define a member name to be created.

MEMBER member-name

(Optional) Names the member to be created, if the data set is a partitioned data set. If the data set is partitioned data set and you do not specify the MEMBER clause, the member name created is TEMPNAME.

REPLACE

(Optional) Indicates that an existing like-named member will be replaced. REPLACE is valid only if you specify the MEMBER clause.

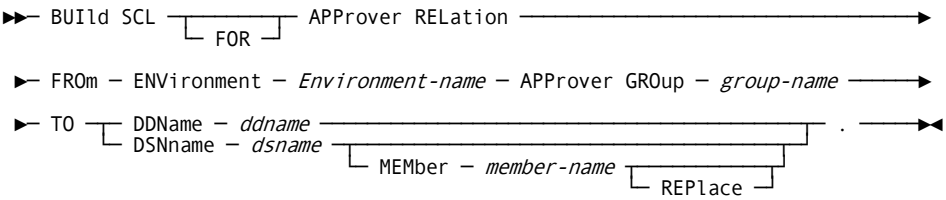
Example: Build SCL for Approver Group SCL

The following is an example of BUILD SCL FOR APPROVER GROUP SCL. The example builds DEFINE SCL using the approver group called ACCTPAY1. The SCL is written to the data set named ENDEVOR.SCLOUT. The member ACCTSPAY replaces any existing like-named member.

```
BUILD SCL FOR APPROVER GROUP ACCTPAY1
        FROM ENVIRONMENT DEVEL
    TO DSNAME "ENDEVOR.SCLOUT"
    MEMBER ACCTSPAY
          REPLACE .
```

Build SCL for Approver Relation Syntax

Use the Build SCL for Approver Relation statement to build DEFINE statements that relate an approver group to a particular inventory area. This SCL statement has the following syntax:



Note: You can use name-masking to specify *some* variable values. A name-masked value is not explicit, because it includes a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. For more information about name-masking, see Name-Masking (see page 23).

BUILD SCL [FOR] APPROVER RELATION

Builds DEFINE statements that relate an approver group to a particular inventory area. The FOR keyword is optional.

FROM ENVIRONMENT *Environment-name* APPROVER GROUP *group-name*

Indicates the approver group and the Environment where that approver group is located.

ENVIRONMENT *Environment-name*

Specifies the Environment location of the approver group from which you are building DEFINE SCL. You can specify a name-masked Environment name.

APPROVER GROUP *group-name*

Specifies the approver group associated with the Environment. You can specify a name-masked approver group name.

TO DDNAME *ddname* | DSNAME *dsname* [MEMBER *member-name* REPLACE]

Indicates where you want the BUILD SCL written to. The data set defined by this clause must be allocated with either fixed or variable length records. If fixed, the record length must be 80. If variable, the record length must be at least 84. Specify either the DDNAME parameter or the DSNAME parameter, but not both.

DDNAME ddname

Identifies the name of an allocated DD statement. The DD statement must define a sequential data set or a partitioned data set with an explicit member.

DSNAME *dsname* [MEMBER *member-name* REPLACE]

Identifies the name of an existing catalogued data set. If the data set name contains imbedded periods, enclose it in quotation marks. If the data set is a partitioned data set, you can use the MEMBER clause to define a member name to be created.

MEMBER member-name

(Optional) Names the member to be created, if the data set is a partitioned data set. If the data set is partitioned data set and you do not specify the MEMBER clause, the member name created is TEMPNAME.

REPLACE

(Optional) Indicates that an existing like-named member will be replaced. REPLACE is valid only if you specify the MEMBER clause.

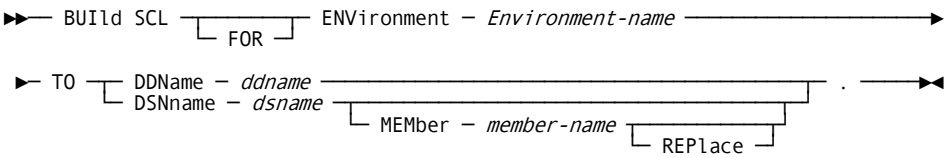
Example: Build SCL for Approver Relation SCL

The following is an example of the BUILD SCL FOR APPROVER RELATION SCL. The example builds DEFINE SCL that relates the ACCTPAY1 approver group to Environment DEVEL. The SCL is written to DD statement SCLOUT.

```
BUILD SCL FOR APPROVER RELATION
          FROM ENVIRONMENT DEVEL
           APPROVER GROUP ACCTPAY1
          TO DDNAME SCLOUT .
```

Build SCL for Environment Syntax

Use the BUILD SCL FOR ENVIRONMENT action to build DEFINE SCL statements for environment definitions. The BUILD SCL FOR ENVIRONMENT action builds DEFINE SCL statements for all inventory definitions (system, subsystem, type, and so on) associated with the environment you specify. However, if the Global Type Sequencing option is enabled at your site, type sequence statements will not be built. This SCL statement has the following syntax:



Note: You can use name-masking to specify *some* variable values. A name-masked value is not explicit, because it includes a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. For more information about name-masking, see Name-Masking (see page 23).

BUILD SCL [FOR] ENVIRONMENT *Environment-name*

Builds DEFINE statements for the specified Environment. You can specify a name-masked environment name. The FOR keyword is optional.

TO DDNAME *ddname* | DSNAME *dsname* [MEMBER *member-name* REPLACE]

Indicates where you want the BUILD SCL written to. The data set defined by this clause must be allocated with either fixed or variable length records. If fixed, the record length must be 80. If variable, the record length must be at least 84. Specify either the DDNAME parameter or the DSNAME parameter, but not both.

DDNAME ddname

Identifies the name of an allocated DD statement. The DD statement must define a sequential data set or a partitioned data set with an explicit member.

DSNAME *dsname* [MEMBER *member-name* REPLACE]

Identifies the name of an existing catalogued data set. If the data set name contains imbedded periods, enclose it in quotation marks. If the data set is a partitioned data set, you can use the MEMBER clause to define a member name to be created.

MEMBER member-name

(Optional) Names the member to be created, if the data set is a partitioned data set. If the data set is partitioned data set and you do not specify the MEMBER clause, the member name created is TEMPNAME.

REPLACE

(Optional) Indicates that an existing like-named member will be replaced. REPLACE is valid only if you specify the MEMBER clause.

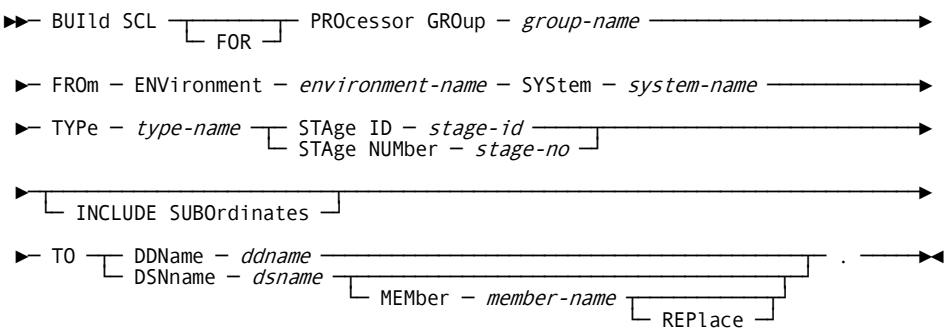
Example: Build SCL for Environment

This example builds DEFINE SCL for an Environment called DEVEL. The SCL is written to the data set named ENDEVOR.SCLOUT. The member DEVEL replaces any existing member named DEVEL.

```
BUILD SCL FOR ENVIRONMENT DEVEL
          TO DSNAME "ENDEVOR.SCLOUT"
                       MEMBER DEVEL
                          REPLACE .
```

Build SCL for Processor Group Syntax

Use the Build SCL For Processor Group statement to build DEFINE statements to set up processor groups. This SCL statement has the following syntax:



Note: You can use name-masking to specify *some* variable values. A name-masked value is not explicit, because it includes a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. For more information about name-masking, see Name-Masking (see page 23).

BUILD SCL [FOR] PROCESSOR GROUP *group-name*

Builds DEFINE statements for the specified processor group. You can specify a name-masked processor group name. The FOR keyword is optional.

FROM ENVIRONMENT environment-name SYSTEM system-name TYPE type-name STAGE ID stage-id | STAGE NUMBER stage-no [INCLUDE SUBORDINATES]

Specifies the inventory location of the processor group. You must specify an environment, system, type, and stage. You can use a name-mask with the environment, system, type, and stage ID. The stage specification can be either one of the following:

STAGE ID

Enter a single alphanumeric stage identifier. Name-masking is supported.

STAGE NUMBER

Enter either 1 or 2. Name-masking is not supported.

INCLUDE SUBORDINATES

(Optional) Create DEFINE SCL for the processor symbols associated with the processor group.

TO DDNAME *ddname* | DSNAME *dsname* [MEMBER *member-name* REPLACE]

Indicates where you want the BUILD SCL written to. The data set defined by this clause must be allocated with either fixed or variable length records. If fixed, the record length must be 80. If variable, the record length must be at least 84. Specify either the DDNAME parameter or the DSNAME parameter, but not both.

DDNAME ddname

Identifies the name of an allocated DD statement. The DD statement must define a sequential data set or a partitioned data set with an explicit member.

DSNAME *dsname* [MEMBER *member-name* REPLACE]

Identifies the name of an existing catalogued data set. If the data set name contains imbedded periods, enclose it in quotation marks. If the data set is a partitioned data set, you can use the MEMBER clause to define a member name to be created.

MEMBER member-name

(Optional) Names the member to be created, if the data set is a partitioned data set. If the data set is partitioned data set and you do not specify the MEMBER clause, the member name created is TEMPNAME.

REPLACE

(Optional) Indicates that an existing like-named member will be replaced. REPLACE is valid only if you specify the MEMBER clause.

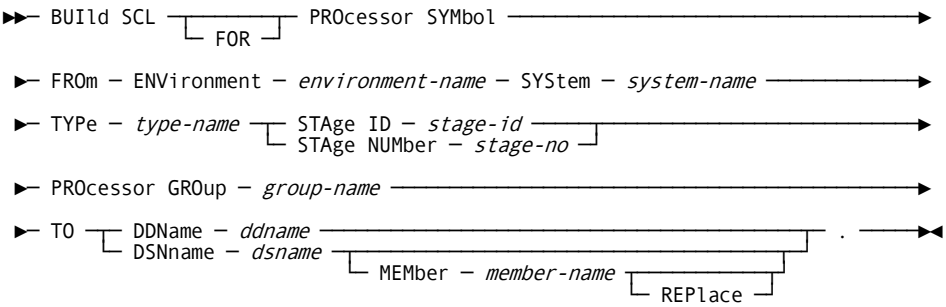
Example: Build SCL for Processor Group SCL

The following is an example of the BUILD SCL FOR PROCESSOR GROUP SCL. The example builds DEFINE SCL for the processor group called COBNBL1. The optional clause INCLUDE SUBORDINATES is specified, therefore, DEFINE SCL is built for all the processor symbols associated with the processor group. The SCL is written to the data set named ENDEVOR.SCLOUT. The member PROCGR1 replaces any existing like-named member.

```
BUILD SCL FOR PROCESSOR GROUP "COBNBL1"
                FROM ENVIRONMENT "DEVEL"
                 SYSTEM "ACCT"
                 TYPE "COBOL"
                 STAGE ID "U"
                  INCLUDE SUBORDINATES
                TO DSNAME "ENDEVOR.SCLOUT"
                MEMBER "PROCGR1"
                        REPLACE .
```

Build SCL for Processor Symbol Syntax

Use the Build SCL For Processor Symbol action to build DEFINE statements to define the processor symbol overrides associated with a processor group. This SCL statement has the following syntax:



Note: You can use name-masking to specify *some* variable values. A name-masked value is not explicit, because it includes a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. For more information about name-masking, see Name-Masking (see page 23).

BUILD SCL FOR PROCESSOR SYMBOL

Builds DEFINE statements for processor symbols. You must specify this clause.

FROM ENVIRONMENT environment-name SYSTEM system-name TYPE type-name STAGE ID stage-id | STAGE NUMBER stage-no PROCESSOR GROUP group-name

Specifies the inventory location of the processor symbols. You must specify an environment, system, type, processor group, and stage. You can use a name-mask with the environment, system, type, processor group, and stage ID. The stage specification can be either one of the following:

STAGE ID

Enter a single alphanumeric stage identifier. Name-masking is supported.

STAGE NUMBER

Enter either 1 or 2. Name-masking is not supported.

TO DDNAME *ddname* | DSNAME *dsname* [MEMBER *member-name* REPLACE]

Indicates where you want the BUILD SCL written to. The data set defined by this clause must be allocated with either fixed or variable length records. If fixed, the record length must be 80. If variable, the record length must be at least 84. Specify either the DDNAME parameter or the DSNAME parameter, but not both.

DDNAME ddname

Identifies the name of an allocated DD statement. The DD statement must define a sequential data set or a partitioned data set with an explicit member.

DSNAME *dsname* [MEMBER *member-name* REPLACE]

Identifies the name of an existing catalogued data set. If the data set name contains imbedded periods, enclose it in quotation marks. If the data set is a partitioned data set, you can use the MEMBER clause to define a member name to be created.

MEMBER member-name

(Optional) Names the member to be created, if the data set is a partitioned data set. If the data set is partitioned data set and you do not specify the MEMBER clause, the member name created is TEMPNAME.

REPLACE

(Optional) Indicates that an existing like-named member will be replaced. REPLACE is valid only if you specify the MEMBER clause.

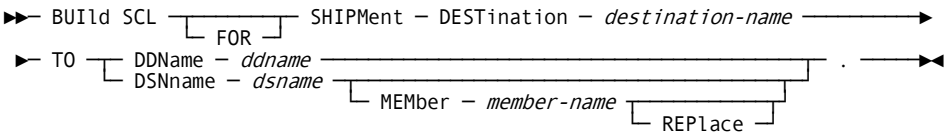
Example: Build SCL for Processor Symbol

This example builds DEFINE SCL for processor symbols using the processor group COBNBL1. DEFINE SCL for all symbols associated with the processors in processor group COBNBL1 is built. The SCL is written to the data set named ENDEVOR.SCLOUT. The member PROCSYM1 replaces any existing like-named member.

```
BUILD SCL FOR PROCESSOR SYMBOL
      FROM ENVIRONMENT "DEVEL"
             SYSTEM "ACCT"
             TYPE "COBOL"
             STAGE ID "U"
             PROCESSOR GROUP "COBNBL1"
     TO DSNAME "ENDEVOR.SCLOUT"
             MEMBER "PROCSYM1"
             REPLACE .
```

Build SCL for Shipment Destination Syntax

Use the Build SCL For Shipment Destination action to build SCL to define a package shipment destination and to define *all* data set and USS mapping rules associated with a package shipment destination. It is not possible to build SCL to define an individual data set or USS mapping rule. This SCL statement has the following syntax:



BUILD SCL [FOR] SHIPMENT DESTINATION *destination-name*

Identifies the one- to seven-character name of the shipment destination from which you are building DEFINE SCL. The FOR keyword is optional.You can specify a name-masked destination name.

TO DDNAME *ddname* | DSNAME *dsname* [MEMBER *member-name* REPLACE]

Indicates where you want the BUILD SCL written to. The data set defined by this clause must be allocated with either fixed or variable length records. If fixed, the record length must be 80. If variable, the record length must be at least 84. Specify either the DDNAME parameter or the DSNAME parameter, but not both.

DDNAME ddname

Identifies the name of an allocated DD statement. The DD statement must define a sequential data set or a partitioned data set with an explicit member.

DSNAME *dsname* [MEMBER *member-name* REPLACE]

Identifies the name of an existing catalogued data set. If the data set name contains imbedded periods, enclose it in quotation marks. If the data set is a partitioned data set, you can use the MEMBER clause to define a member name to be created.

MEMBER member-name

(Optional) Names the member to be created, if the data set is a partitioned data set. If the data set is partitioned data set and you do not specify the MEMBER clause, the member name created is TEMPNAME.

REPLACE

(Optional) Indicates that an existing like-named member will be replaced. REPLACE is valid only if you specify the MEMBER clause.

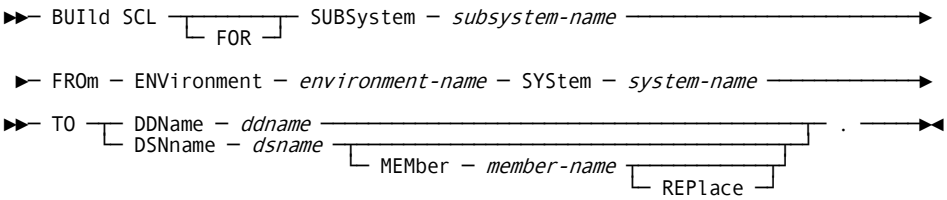
Example: Build SCL for Shipment Destination SCL

The following is an example of the BUILD SCL FOR SHIPMENT DESTINATION SCL. The example builds DEFINE SCL for a shipment destination called DEST001. The SCL is written to the DD statement SCLOUT.

```
BUILD SCL FOR SHIPMENT DESTINATION "DEST001"
                  TO DDNAME "SCLOUT" .
```

Build SCL for Subsystem Syntax

Use the Build SCL For Subsystem action to build DEFINE statements for subsystem definitions. This SCL statement has the following syntax:



Note: You can use name-masking to specify *some* variable values. A name-masked value is not explicit, because it includes a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. For more information about name-masking, see Name-Masking (see page 23).

BUILD SCL[FOR] SUBSYSTEM *subsystem-name*

Identifies the 1- to 8-character name of the subsystem from which you are building DEFINE SCL. The FOR keyword is optional. You can specify a name-masked subsystem name.

FROM ENVIRONMENT *environment-name*
SYSTEM *system-name*

Identifies the inventory location of the subsystem from which you are building DEFINE SCL. You must also enter the name of the system to which the subsystem is defined. You can use name-masking to specify the environment name and the system name.

TO DDNAME *ddname* | DSNAME *dsname* [MEMBER *member-name* REPLACE]

Indicates where you want the BUILD SCL written to. The data set defined by this clause must be allocated with either fixed or variable length records. If fixed, the record length must be 80. If variable, the record length must be at least 84. Specify either the DDNAME parameter or the DSNAME parameter, but not both.

DDNAME ddname

Identifies the name of an allocated DD statement. The DD statement must define a sequential data set or a partitioned data set with an explicit member.

DSNAME *dsname* [MEMBER *member-name* REPLACE]

Identifies the name of an existing catalogued data set. If the data set name contains imbedded periods, enclose it in quotation marks. If the data set is a partitioned data set, you can use the MEMBER clause to define a member name to be created.

MEMBER member-name

(Optional) Names the member to be created, if the data set is a partitioned data set. If the data set is partitioned data set and you do not specify the MEMBER clause, the member name created is TEMPNAME.

REPLACE

(Optional) Indicates that an existing like-named member will be replaced. REPLACE is valid only if you specify the MEMBER clause.

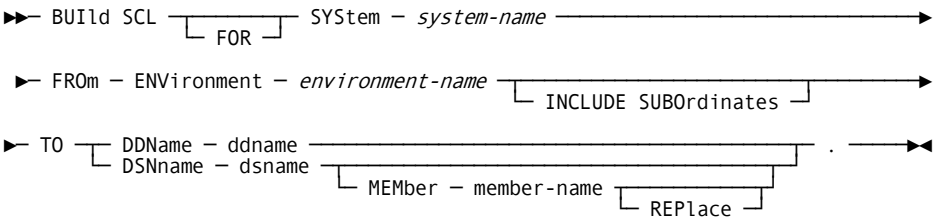
Example: Build SCL for Subsystem SCL

The following is an example of the BUILD SCL FOR SUBSYSTEM SCL. The example builds DEFINE SCL for all subsystems using environment DEVEL and system ACCT. The SCL is written to the DD statement SCLOUT.

```
BUILD SCL FOR SUBSYSTEM "*"
    FROM ENVIRONMENT "DEVEL"
         SYSTEM "ACCT"
    TO DDNAME "SCLOUT" .
```

Build SCL for System Syntax

Use the Build SCL For System action to build DEFINE statements for system definitions. However, if the Global Type Sequencing option is enabled at your site, type sequence statements will not be built. This SCL statement has the following syntax:



Note: You can use name-masking to specify *some* variable values. A name-masked value is not explicit, because it includes a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. For more information about name-masking, see Name-Masking (see page 23).

BUILD SCL [FOR] SYSTEM *system-name*

Identifies the one- to eight-character name of the system from which you wish to build DEFINE SCL. The FOR keyword is optional. You can specify a name-masked system name.

FROM ENVIRONMENT *environment-name* [INCLUDE SUBORDINATES]

Identifies the environment location of the system from which you are building DEFINE SCL. You can specify a name-masked environment name.

INCLUDE SUBORDINATES

(Optional) Create DEFINE SCL for the subsystem, type, type sequence, processor group, and processor group symbolic definitions associated with this system.

TO DDNAME *ddname* | DSNAME *dsname* [MEMBER *member-name* REPLACE]

Indicates where you want the BUILD SCL written to. The data set defined by this clause must be allocated with either fixed or variable length records. If fixed, the record length must be 80. If variable, the record length must be at least 84. Specify either the DDNAME parameter or the DSNAME parameter, but not both.

DDNAME ddname

Identifies the name of an allocated DD statement. The DD statement must define a sequential data set or a partitioned data set with an explicit member.

DSNAME *dsname* [MEMBER *member-name* REPLACE]

Identifies the name of an existing catalogued data set. If the data set name contains imbedded periods, enclose it in quotation marks. If the data set is a partitioned data set, you can use the MEMBER clause to define a member name to be created.

MEMBER member-name

(Optional) Names the member to be created, if the data set is a partitioned data set. If the data set is partitioned data set and you do not specify the MEMBER clause, the member name created is TEMPNAME.

REPLACE

(Optional) Indicates that an existing like-named member will be replaced. REPLACE is valid only if you specify the MEMBER clause.

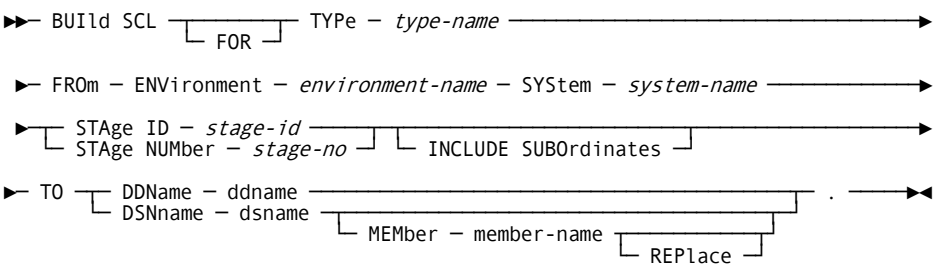
Example: Build SCL for System SCL

The following is an example of the BUILD SCL FOR SYSTEM SCL. The example builds DEFINE SCL for the system ACCT using environment DEVEL. The optional clause INCLUDE SUBORDINATES is specified, therefore, DEFINE SCL is built for all inventory definitions associated with system ACCT. The SCL is written to the data set named ENDEVOR.SCLOUT. The member SYSACCT replaces any existing like-named member.

```
BUILD SCL FOR SYSTEM "ACCT"
    FROM ENVIRONMENT "DEVEL"
    INCLUDE SUBORDINATES
    TO DSNAME "ENDEVOR.SCLOUT"
        MEMBER "SYSACCT"
                 REPLACE .
```

Build SCL for Type Syntax

Use the BUILD SCL FOR TYPE action to build DEFINE SCL statements for type definitions. This SCL statement has the following syntax:



Note: You can use name-masking to specify *some* variable values. A name-masked value is not explicit, because it includes a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. For more information about name-masking, see Name-Masking (see page 23).

BUILD SCL [FOR] TYPE *type-name*

Specifies the one- to eight-character name of the type from which you are building DEFINE SCL. The FOR keyword is optional. You can specify a name-masked type name.

FROM ENVIRONMENT environment-name SYSTEM system-name STAGE ID stage-id | STAGE NUMBER stage-no [INCLUDE SUBORDINATES]

Specifies the inventory location of the type. You must specify an environment, system, and stage. You can use a name-mask with the environment, system, type, and stage ID. The stage specification can be either one of the following:

- STAGE ID— Enter a single alphanumeric stage identifier. Name-masking is supported.
- STAGE NUMBER— Enter either 1 or 2. Name-masking is *not* supported.

INCLUDE SUBORDINATES

(Optional) Creates DEFINE SCL for the processor group and processor group symbol definitions associated with the type.

TO DDNAME *ddname* | DSNAME *dsname* [MEMBER *member-name* REPLACE]

Indicates where you want the BUILD SCL written to. The data set defined by this clause must be allocated with either fixed or variable length records. If fixed, the record length must be 80. If variable, the record length must be at least 84. Specify either the DDNAME parameter or the DSNAME parameter, but not both.

DDNAME ddname

Identifies the name of an allocated DD statement. The DD statement must define a sequential data set or a partitioned data set with an explicit member.

DSNAME *dsname* [MEMBER *member-name* REPLACE]

Identifies the name of an existing catalogued data set. If the data set name contains imbedded periods, enclose it in quotation marks. If the data set is a partitioned data set, you can use the MEMBER clause to define a member name to be created.

MEMBER member-name

(Optional) Names the member to be created, if the data set is a partitioned data set. If the data set is partitioned data set and you do not specify the MEMBER clause, the member name created is TEMPNAME.

REPLACE

(Optional) Indicates that an existing like-named member will be replaced. REPLACE is valid only if you specify the MEMBER clause.

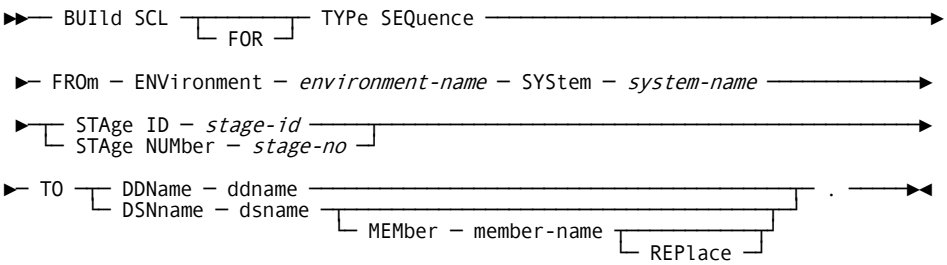
Example: Build SCL for Type SCL

The following is an example of the BUILD SCL FOR TYPE SCL. The example builds DEFINE SCL for all types using environment DEVEL, system ACCT, and stage number 1. The SCL is written to the DD statement SCLOUT.

```
BUILD SCL FOR TYPE "*"
                FROM ENVIRONMENT "DEVEL"
                           SYSTEM "ACCT"
                           STAGE NUMBER 1
                TO DDNAME "SCLOUT" .
```

Build SCL for Type Sequence Syntax

Use the Build SCL For Type Sequence action to build DEFINE statements for type sequence definitions. The Build SCL For Type Sequence action assigns sequence numbers beginning with 5 and incremented by 10 for each type. If the Global Type Sequencing option is enabled at your site, the statement will be rejected as an error.This SCL statement has the following syntax:



Note: You can use name-masking to specify *some* variable values. A name-masked value is not explicit, because it includes a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. For more information about name-masking, see Name-Masking (see page 23).

BUILD SCL FOR TYPE SEQUENCE

Create DEFINE SCL from an existing type sequence definition. You must specify this clause.

FROM ENVIRONMENT environment-name SYSTEM system-name STAGE ID stage-id | STAGE NUMBER stage-no

Specifies the inventory location of the type sequence. You must specify an environment, system, and stage. You can use a name-mask with the environment, system, type, and stage ID. The stage specification can be either one of the following:

- STAGE ID— Enter a single alphanumeric stage identifier. Name-masking is supported.
- STAGE NUMBER— Enter either 1 or 2. Name-masking is *not* supported.

TO DDNAME *ddname* | DSNAME *dsname* [MEMBER *member-name* REPLACE]

Indicates where you want the BUILD SCL written to. The data set defined by this clause must be allocated with either fixed or variable length records. If fixed, the record length must be 80. If variable, the record length must be at least 84. Specify either the DDNAME parameter or the DSNAME parameter, but not both.

DDNAME ddname

Identifies the name of an allocated DD statement. The DD statement must define a sequential data set or a partitioned data set with an explicit member.

DSNAME *dsname* [MEMBER *member-name* REPLACE]

Identifies the name of an existing catalogued data set. If the data set name contains imbedded periods, enclose it in quotation marks. If the data set is a partitioned data set, you can use the MEMBER clause to define a member name to be created.

MEMBER member-name

(Optional) Names the member to be created, if the data set is a partitioned data set. If the data set is partitioned data set and you do not specify the MEMBER clause, the member name created is TEMPNAME.

REPLACE

(Optional) Indicates that an existing like-named member will be replaced. REPLACE is valid only if you specify the MEMBER clause.

Example: Build SCL for Type Sequence SCL

The following is an example of the BUILD SCL FOR TYPE SEQUENCE SCL. The example builds DEFINE SCL for a type sequence using environment DEVEL, system ACCT, and stage ID U. The SCL is written to the data set named ENDEVOR.SCLOUT. The member ACCTSEQ1 replaces any existing like-named member.

```
BUILD SCL FOR TYPE SEQUENCE
        FROM ENVIRONMENT "DEVEL"
         SYSTEM "ACCT"
         STAGE ID "U"
    TO DSNAME "ENDEVOR.SCLOUT"
         MEMBER "ACCTSEQ1"
           REPLACE .
```

Define Statements

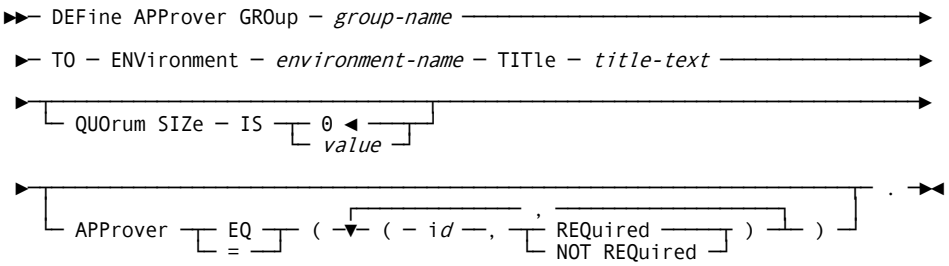
Use DEFINE statements to create or update environment definitions. An environment definition is created if it does not exist. An update occurs if the environment definition exists.

The following general conventions apply to all DEFINE statements:

- Names you specify in the DEFINE clause can have a maximum of 8 characters with the exception of SHIPMENT DESTINATION, which can be no longer than 7 characters, and APPROVER GROUP, which can be no longer than 16 characters.
- You can use partially or fully wildcarded names in DEFINE clauses. Wildcarded names indicate that an update is to be performed on existing environment definitions that match the name you specify.
- You cannot specify names that include imbedded spaces, non-alphabetical, non-numeric, or non-national characters. However, this does not fully apply to the PROCESSOR OUTPUT TYPE, which can contain any characters or imbedded spaces, but cannot start with a leading blank.
- Updating an inventory definition does not change settings that are not specified in the Define statement used to update the inventory definition. For example, after the first DEFINE statement creates the inventory definition, the second DEFINE statement only updates the settings specified in the second DEFINE statement. Any settings not specified in the second DEFINE statement are not changed.
- Variable values can be enclosed in double or single quotes, but quotes are not required, except for data set names. All variable values on Define statements created by Build statements are enclosed in quotes by default.

Define Approver Group Syntax

Use the DEFINE APPROVER GROUP action to create or update approver group definitions. This SCL statement has the following syntax:



DEFINE APPROVER GROUP group-name

The DEFINE APPROVER GROUP clause identifies the up to sixteen-character name of the approver group you are creating or updating. You can specify a partially or fully wildcarded approver group name. A wildcarded approver group name updates all matching approver group definitions.

TO ENVIRONMENT environment-name

The TO ENVIRONMENT clause identifies the environment to which you are defining the approver group. You must use a fully specified and non-wildcarded environment name.

TITLE title-text

The TITLE clause identifies a 1- to 50-character description of the approver group you are creating or updating. You must specify the TITLE clause if are creating an approver group. The clause is optional if you are updating an approver group. If the text contains imbedded spaces, enclose it in single or double quotation marks.

QUORUM SIZE IS value

The QUORUM SIZE CLAUSE IS clause specifies the minimum number of approvers who must approve a package before it can be executed. The quorum size is optional when both creating and updating an approver group definition. If you specify this clause, the value must between 0 and 16. If you do not specify this clause, the default is 0.

APPROVER EQ = id, REQUIRED/NOT REQUIRED

The APPROVER clause specifies whether the approver ID is required to approve the package. The clause contains two fields, the approver ID field and the REQUIRED or NOT REQUIRED field. If you specify more than one approver ID, enclose the fields in parentheses and separate by commas. For example, to define IDs USER01 and USER02 as required approvers, the APPROVER clause would be specified as:

```
APPROVER=((USER01,REQUIRED),(USER02,REQUIRED))
```

When creating or updating an Approver Group definition, the APPROVER clause is optional. If an Approver Group has no approvers, CA Endevor SCM uses external approvers and uses the approver group name as the external security group or profile name.

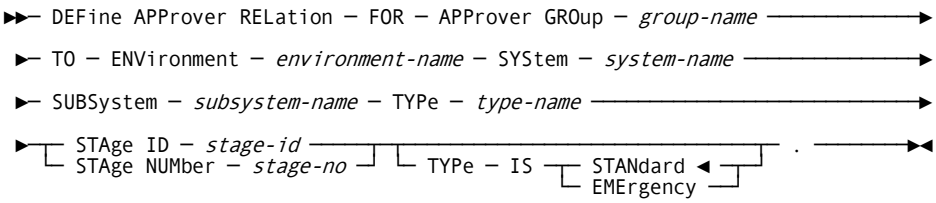
Example: Define Approver Group SCL

The following is an example of the DEFINE APPROVER GROUP SCL. The example creates an Approver Group named ACCTPAY. This group contains five User IDs. Two of the five User IDs are required for approval. There is a quorum size of three which means that one of the not required users must also approve the package.

```
DEFINE APPROVER GROUP "ACCTPAY1"
          TO ENVIRONMENT "DEVEL"
          TITLE "Accounts Payable Approver Group"
          QUORUM SIZE IS 3
          APPROVER EQ ( (USER001, REQUIRED),
                        (USER002, NOT REQUIRED),
                        (USER003, NOT REQUIRED),
                        (USER004, REQUIRED),
                        (USER005, NOT REQUIRED) ) .
```

Define Approver Relation Syntax

Use the DEFINE APPROVER RELATION action to create a new approver relation definition. It is not possible to update an existing approver relation definition. This SCL statement has the following syntax:



DEFINE APPROVER RELATION

The DEFINE APPROVER RELATION clause indicates that you are creating a new approver relation definition. You must specify this clause.

FOR APPROVER GROUP group-name

The FOR APPROVER GROUP clause identifies the 1- to 16-character name of the approver group for which an inventory relationship is being built. You must use a fully specified and non-wildcarded approver group name.

TO **ENVIRONMENT environment-name**
SYSTEM system-name
SUBSYSTEM subsystem-name
TYPE type-name
STAGE ID stage-id
STAGE NUMBER stage-no

The TO clause identifies the inventory location to which you are defining the approver relation. You must specify a fully qualified environment name.

The TO SYSTEM, SUBSYSTEM, TYPE, and STAGE clauses you specify can be either fully qualified or fully wildcarded. You cannot use partially wildcarded names.

You must use the STAGE ID clause if you wild card the stage. Values specified in a wildcarded STAGE clause are not expanded. The approver group relationship is built with the wildcarded values.

TYPE IS STANDARD/EMERGENCY

The TYPE IS STANDARD/EMERGENCY clause specifies the approver type for this approver group. You must specify this clause when creating an approver group relation definition.

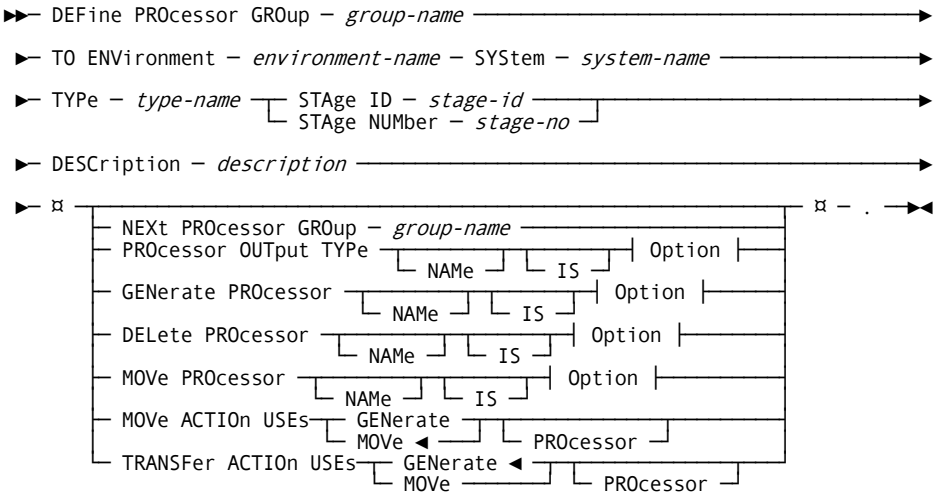
Example: Define Approver Relation SCL

The following is an example of the DEFINE APPROVER RELATION SCL. The example creates an Approver Relation for Approver Group ACCTPAY. This Approver Relation is for the environment DEVEL, system ACCT, subsystem ACCTPAY, type COBOL, stage number 1.

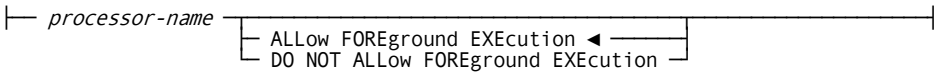
```
DEFINE APPROVER RELATION
            FOR APPROVER GROUP "ACCTPAY"
       TO ENVIRONMENT "DEVEL"
          SYSTEM "ACCT"
          SUBSYSTEM "ACCTPAY1"
          TYPE "COBOL"
                  STAGE NUMBER 1
                  TYPE IS STANDARD .
```

Define Processor Group Syntax

Use the DEFINE PROCESSOR GROUP action to create a new or update an existing processor group definition. This SCL statement has the following syntax:



Expansion of Option



DEFINE PROCESSOR GROUP group-name

The DEFINE PROCESSOR GROUP clause identifies the 1- to 8-character name of the processor group you are creating or updating. The can specify a partially or fully wildcarded processor group name. A wildcarded processor group name updates all matching processor group definitions.

TO **ENVIRONMENT environment-name**
SYSTEM system-name
TYPE type-name
STAGE ID stage-id
STAGE NUMBER stage-no

The TO clause identifies the inventory location to which the processor group is defined or is to be defined. Names you specify in the TO clause must all be fully specified.

DESCRIPTION description

Use the DESCRIPTION clause to enter text of up to 50 characters in length describing the processor group. If the text contains embedded spaces, enclose it in either single or double quotation marks. The description clause is required when creating a processor group definition and optional when updating a processor group definition.

NEXT PROCESSOR GROUP group-name

The NEXT PROCESSOR GROUP clause identifies the name of the processor group at the next map location. If you do not specify the NEXT PROCESSOR GROUP clause, the clause defaults to the name of the processor group that you are defining.

PROCESSOR OUTPUT TYPE

The PROCESSOR OUTPUT TYPE designates the kind of output in this processor group. The character default of 16 is concatenation of type names and processor group names. This is used with the element registration feature and can be user defined. Any string of 16 characters is allowed; however, the string cannot start with leading spaces and must be contained within double quotes.

GENERATE/DELETE/MOVE PROCESSOR NAME processor-name

The GENERATE/DELETE/MOVE PROCESSOR NAME clauses identifies a one- to-eight character alpha-numeric name of the processors that make up the processor group. The GENERATE/MOVE/DELETE PROCESSOR NAME clauses are optional when creating or updating a processor group definition.

If you do not specify the clause, the processor name defaults to *NOPROC*. If the processor name identifies one of CA Endevor SCM's reserved processor names (GPPROCSS, DPPROCSS, BASICGEN or BASICDEL), the processor name converts to *NOPROC*. You cannot use the clauses MOVE ACTION USES or TRANSFER ACTION USES with *NOPROC*.

MOVE ACTION USES GENERATE/MOVE PROCESSOR

The MOVE ACTION USES GENERATE/MOVE PROCESSOR clause indicates whether CA Endevor SCM is to execute the generate or the move processor as part of the MOVE action. You cannot use this clause with a processor name of *NOPROC*.

TRANSFER ACTION USES GENERATE/MOVE PROCESSOR

The TRANSFER ACTION USES GENERATE/MOVE PROCESSOR clause indicates whether CA Endevor SCM is to execute the generate or the move processor as part of the TRANSFER action. You cannot use this clause with a processor name of *NOPROC*.

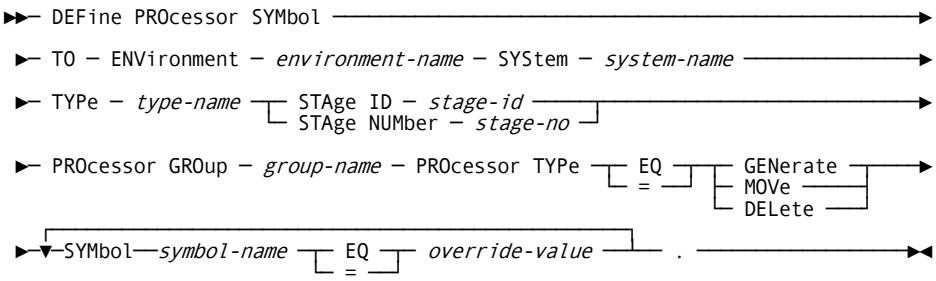
Example: Define Processor Group SCL

The following is an example of the DEFINE PROCESSOR GROUP SCL. The example updates processor group COBNBL1. It updates the transfer action so that the TRANSFER action uses the move processor instead of the generate processor.

```
DEFINE PROCESSOR GROUP "COBNBL1"
      TO ENVIRONMENT "DEVEL"
                 SYSTEM "ACCT"
                 TYPE "COBOL"
                 STAGE ID "U"
            TRANSFER ACTION USES MOVE PROCESSOR .
```

Define Processor Symbol Syntax

Use the DEFINE PROCESSOR SYMBOL action to define or update symbols in processors.This SCL statement has the following syntax:



DEFINE PROCESSOR SYMBOL

The DEFINE PROCESSOR SYMBOL clause indicates that you are to define or update symbols in processors. You must specify this clause.

TO **ENVIRONMENT environment-name**
SYSTEM system-name
TYPE type-name
STAGE ID stage-id
STAGE NUMBER stage-no
PROCESSOR GROUP group-name
PROCESSOR TYPE EQ/= GENERATE/MOVE/DELETE

The TO clause identifies the inventory location of the processor group to which the processor symbols are defined or are to be defined, the processor group, and a processor type within the group.

You must fully specify the environment name, system name, and type name.

You can fully specify, partially wildcard or fully wildcard the processor group name. A wildcarded processor group name updates matching processor symbolic definitions. The processor group name cannot be '*NOPROC*'. Specify either a generate, move, or delete processor type.

SYMBOL symbol-name EQ/= override-value

The SYMBOL clause identifies the one- to eight-character name of the processor symbol you are modifying. The symbol must be defined in the processor.

The override value identifies the up to 65-character override value to be associated with the symbol. If the override value contains imbedded single quotation marks enclose the field in double quotation marks. Likewise, if it contains imbedded double quotation marks enclose it in single quotation marks. The override value cannot contain both single and double quotation marks. You can specify multiple symbolic values by repeating the SYMBOL clause as many times as needed.

Example: Define Processor Symbol SCL

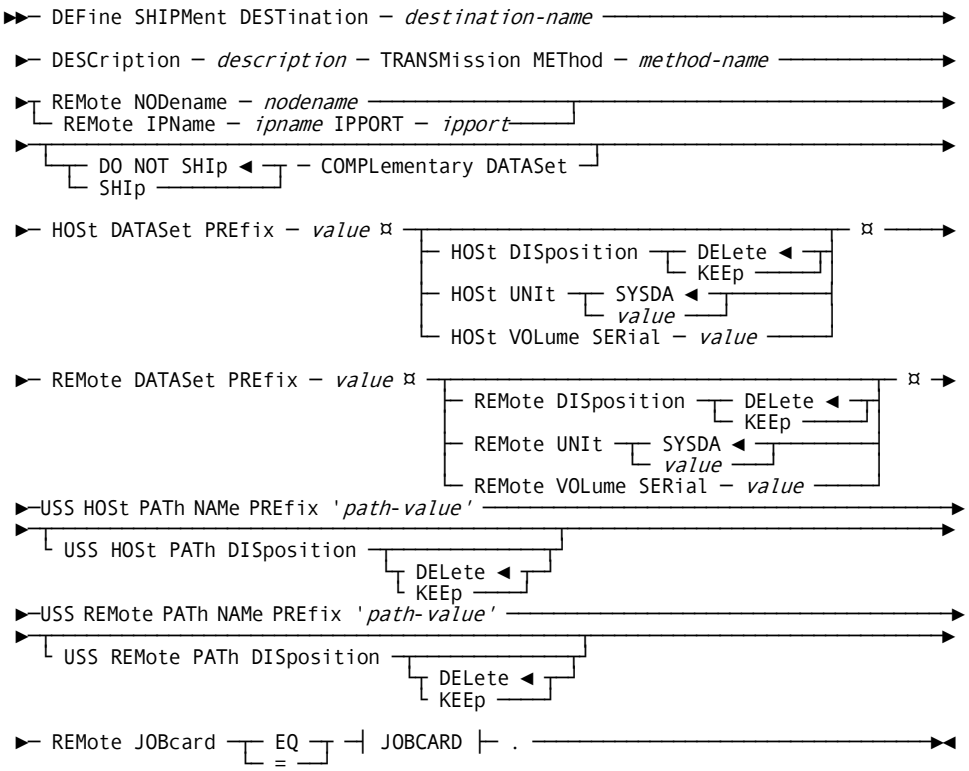
The following is an example of the DEFINE PROCESSOR SYMBOL SCL. The example updates processor symbols for processor group COBNBL1. The symbols for the generate processor are updated.

```
DEFINE PROCESSOR SYMBOL
    TO  ENVIRONMENT "DEVEL"
                SYSTEM "ACCT"
                TYPE "COBOL"
                STAGE ID "U"
                PROCESSOR GROUP "COBNBL1"
                PROCESSOR TYPE EQ GENERATE
    SYMBOL SYSOUT EQ "A"
    SYMBOL PARMCOB EQ "NOLIB, LANGLVL(1)"
    SYMBOL WRKUNIT EQ "SYSDA"
    SYMBOL EXPINC EQ "N" .
```

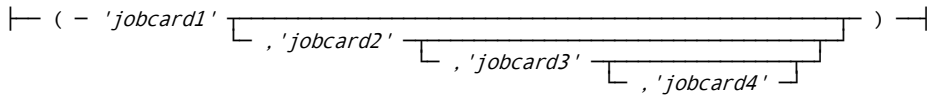
The Define Shipment Destination Action

Use the DEFINE SHIPMENT DESTINATION action to define or update destinations to which you ship package outputs.

Define Shipment Destination Syntax



Expansion of JOBCARD



DEFINE SHIPMENT DESTINATION *destination-name*

Identifies the one- to seven-character name of the destination you are creating or updating. The destination name you specify must be a fully qualified value.

DESCRIPTION *description*

Describes the destination. A valid value is up to 30 characters in length. If the text contains embedded spaces, enclose it in either single or double quotation marks. You must specify this clause when creating a shipment destination. It is optional when updating a shipment destination.

Important! You must specify the TRANSMISSION METHOD clause when creating a package shipment destination. The clause is optional when updating a package shipment destination.

TRANSMISSION METHOD *method-name*

Defines the transmission method that is to be used to ship the package to the remote destination. The following transmission methods are valid, but you must specify the method that corresponds to the transmission utility enabled at your site.

BDT

Specifies the transmission method for the transmission utility Bulk Data Transfer version 2 or above.

BDTNJE

Specifies the transmission method for the transmission utility Bulk Data Transfer version 1 (NJE).

LOCAL

Specifies the transmission method for the transmission utility IEBCOPY.

NDM

Specifies the transmission method for the transmission utility Network DataMover

NETVIEWFTP

Specifies the transmission method for the transmission utility NetView File Transfer Program

XCOM

Specifies the transmission method for the transmission utility XCOM

Note: For the LOCAL and BDTNJE transmission methods, the REMOTE NODENAME and IPNAME are ignored and, therefore, are not required for those transmission methods.

REMOTE NODENAME nodename

Specifies the 1-to 16-character site name to which package outputs are to be shipped. You must specify the REMOTE NODENAME clause when creating a shipment destination. The clause is optional when updating a shipment destination. For the XCOM transmission method, you can specify the VTAM NODENAME or a TCP/IP IPNAME address, but not both.

REMote IPName *ipname* IPPort *ipport*

Specifies the TCP/IP address to which the package outputs are to be shipped. This option is only valid with the XCOM transmission method. For the XCOM transmission method, you can specify a REMOTE NODENAME or a REMOTE IPNAME, but not both. If IPNAME is specified, then IPPORT is required.

IPName *ipname*

Specifies the 1- to 63- character TCP/IP IPNAME shipment destination address for a shipment performed using the XCOM transmission method. If the value contains special characters, which it usually does, it must be enclosed in single quotes. If IPNAME is specified, then IPPORT is required.

IPPort *ipport*

Specifies the one to five character IPPORT number within the IPNAME address. The IPPORT value must be in the range of 1 to 65535. This parameter must match the XCOM SERVPORT specification of the target server. If IPNAME is specified, then IPPORT is required.

SHIP/DO NOT SHIP COMPLEMENTARY DATASET

Indicates whether or not data sets can be shipped along with package shipments. The default is DO NOT SHIP COMPLEMENTARY DATASET.

HOST DATASET PREFIX value

Defines the 1 to 14 character prefix to be assigned to the staging data sets that are created at the host node. You must specify the HOST DATASET PREFIX clause when creating a package shipment destination. The clause is optional when updating a package shipment destination.

HOST DISPOSITION DELETE/KEEP

Specifies the disposition of the host staging data sets after the package shipment utility is complete. The default disposition is DELETE.

HOST UNIT SYSDA/value

Specifies the 1 to 8 character alpha-numeric unit type on which the staging data set is allocated. The default value is SYSDA. The utility does *not* verify that the value you specify is defined to the system.

HOST VOLUME SERIAL value

Identifies the 1 to 6 character volume on which the host staging data sets will be allocated. The utility does *not* verify that the volume serial you specify is defined to the system.

Note: For the LOCAL transmission method, the REMOTE DATASET PREFIX, DISPOSITION, UNIT AND VOLUME SERIAL are ignored if coded and, therefore, are not required for that transmission method.

REMOTE DATASET PREFIX value

Defines the prefix to be assigned to the staging data sets that are created at the remote node. It can be any number of data set name qualifiers of up to 14 characters in length. The clause is required when creating a package shipment destination and optional when updating a package shipment destination.

REMOTE DISPOSITION DELETE/KEEP

Specifies the disposition of the remote staging data sets after the package shipment utility is complete. The default disposition is DELETE.

REMOTE UNIT SYSDA/value

Specifies the unit type on which the staging data set will be allocated. The default value is SYSDA. You can use any string of up to eight alphanumeric characters. The utility does *not* verify that the value you enter is defined to the system.

REMOTE VOLUME SERIAL value

Identifies the volume on which the remote staging data sets will be allocated. You can use any alphanumeric string of up to six characters in length. The utility does *not* verify that the volume serial you enter is defined to the system.

USS HOST PATH NAME PREFIX *path-value*

Assigns this prefix to the staging USS files that are created at the host node.

When creating a package destination, if this clause is not specified, a default prefix of /tmp/ is defined for USS transmission types (LOCAL, NDM,XCOM). This clause is not valid for non-USS transmission types (BDT,BDTNJE,NETVIEWFTP).

When updating a package destination, if the transmission type changes from USS to non-USS, the package record's USS HOST PATH NAME PREFIX length field is set to zero and the name is initialized to binary zero. A warning message and the original prefix value are printed when this occurs.

When a path name value is assigned to the definition, the prefix path name field is padded with blanks.

Valid path-values meet the following requirements:

- Can be up to 732 characters.
- Must start and end with a forward slash (/). If the starting character is not a slash, a syntax error is issued. If the last character is not a slash, a slash is generated.
- If the path name contains a single quote character, use double quotes to enclose the text. A path name cannot contain both a single and a double quote character.
- If the path name cannot fit on a single statement, the overflow may be specified on subsequent lines. Enclose the first string in single quotes then follow it with a comma to indication that an additional line text exists. The pathname specification can not exceed 14 text string values.

USS HOST PATH DISPOSITION DELETE/KEEP

Specifies the disposition of the USS host staging files after the package shipment utility is complete. The default disposition is DELETE. This clause is not valid for non-USS transmission types (BDT,BDTNJE, NETVIEWFTP).

On an update, if the transmission type changes from USS to non-USS, the package record's USS disposition field is set to blanks.

USS REMOTE PATH NAME PREFIX *path-value*

Assigns this prefix to the staging USS files that are created at the remote node.

When creating a package destination, if this clause is not specified, a default prefix of /tmp/ is defined for USS transmission types (LOCAL, NDM,XCOM). This clause is not valid for non-USS transmission types (BDT,BDTNJE,NETVIEWFTP).

When updating a package destination, if the transmission type changes from USS to non-USS, the package record's USS REMOTE PATH NAME PREFIX length field is set to zero and the name is initialized to binary zero. A warning message and the original value are printed when this occurs.

When a path name value is assigned to the definition, the prefix path name field is padded with blanks.

Valid path-values meet the following requirements:

- Can be up to 732 characters.
- Must start and end with a forward slash (/). If the starting character is not a slash, a syntax error is issued. If the last character is not a slash, a slash is generated.
- If the path name contains a single quote character, use double quotes to enclose the text. A path name cannot contain both a single and a double quote character.
- If the path name cannot fit on a single statement, the overflow may be specified on subsequent lines. Enclose the first string in single quotes then follow it with a comma to indication that an additional line text exists. The pathname specification can not exceed 14 text string values.

USS REMOTE PATH DISPOSITION DELETE/KEEP

Specifies the disposition of the USS remote staging files after the package shipment utility is complete. The default disposition is DELETE. This clause is not valid for non-USS transmission types (BDT,BDTNJE, NETVIEWFTP).

On an update, if the transmission type changes from USS to non-USS, the package record's USS disposition field is set to blanks.

REMOTE JOBCARD EQ/= *jobcard*

Specifies the JCL jobcard to be used at the remote site. If the jobcard statement contains a single quotation mark enclose the entire jobcard in double quotation marks. Each jobcard can be no longer than 65 characters, excluding the delimiting quotation marks. The REMOTE JOBCARD clause is required when creating a package shipment destination and is optional when updating a package shipment destination.

Example: USS Path Names in Define Shipment Destination Statements

The path-names in the following clauses are properly specified:

```
USS REMOTE PATH NAME PREFIX  '/u/users/endevor',
    '/stg/'
USS REMOTE PATH NAME PREFIX '/u/users/endevor/',
    "dir'stg1/" ,
    'dirastg2/'
```

The following example shows an invalid path name specification, because it contains both a single and double quote character:

```
USS REMOTE PATH NAME PREFIX '/u/users/endevor/',
    "dir'stg1/" ,
    'dir"stg2/'
```

Example: Define Destination Shipment SCL for Network DataMover

The following is an example of the DEFINE SHIPMENT DESTINATION SCL. The example creates a shipment destination named BOSTNDM. The transmission method is Network DataMover (NDM).

CREATE a Shipment Destination named BOSTNDM. The Transmission Method will be NDM (Network DataMover).

```
DEFINE SHIPMENT DESTINATION 'BOSTNDM'
                DESCRIPTION 'BOSTON DATA MOVER NODE'
                TRANSMISSION METHOD 'NDM'
                REMOTE NODENAME 'CHINNDM'
                DO NOT SHIP COMPLEMENTARY DATASET
                HOST DATASET PREFIX 'USER01.NDVR'
                HOST DISPOSITION DELETE
                HOST UNIT SYSDA
                REMOTE DATASET PREFIX 'USER01.NDVR'
                REMOTE DISPOSITION DELETE
                REMOTE UNIT SYSDA
                REMOTE JOBCARD =
 ("//JOBNAME JOB (ACCOUNT),'JOHN DOE'") ,
 "//*" ,
 "//*" ,
 "//*" )
```

Example: Define Destination Shipment SCL for XCOM Transmission

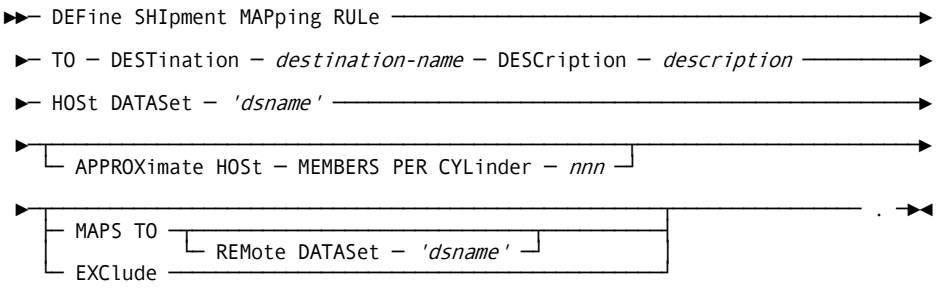
This Define Destination Shipment SCL defines a shipment destination named JPMXCOM for the XCOM transmission method. The REMOTE IPNAME and IPPORT define the TCP/IP address to which the package outputs will be shipped.

```
DEFINE SHIPMENT DESTINATION 'JPMXCOM'
                DESCRIPTION 'LOCAL XCOM TO AB31'
                TRANSMISSION METHOD 'XCOM'
                REMOTE NODENAME 'XCPS'
                REMOTE IPNAME
'VERY.LONG.REMOTE.IPNAME.AAAA.BBBB.CCCC.DDDD.EEEE.FFFF.GGGG.23'
                IPPORT 8044
                HOST DATASET PREFIX 'PUBLIC'
                HOST DISPOSITION DELETE
                REMOTE DATASET PREFIX 'PUBLIC'
                REMOTE DISPOSITION DELETE
                REMOTE JOBCARD EQ
 ("//USER02S JOB 108300000,PSHIPXCM,USER=USERR02,",
 "//  REGION=6M,MSGCLASS=X,NOTIFY=&SYSUID        ",
 "/*JOBPARM SYSAFF=AB31                          ",
 "//*                                           " )
```

The Define Shipment Mapping Rule Action

Use the DEFINE SHIPMENT MAPPING RULE action to create or update mapping rules between a host data set name and a remote data set name.

Define Shipment Mapping Rule Syntax



DEFINE SHIPMENT MAPPING RULE

The DEFINE SHIPMENT MAPPING RULE indicates that you are creating or updating a mapping rule between a host data set name and a remote data set name. You must specify this clause.

TO DESTINATION destination-name

The TO DESTINATION clause identifies the 1- to 7-character name of an existing package shipment destination. You must specify a fully specified value.

DESCRIPTION description

Use the DESCRIPTION clause to enter text of up to 40 characters in length describing this data set map. If the text you enter contains embedded spaces enclose it in either single or double quotation marks. You must specify the DESCRIPTION clause when creating a shipment mapping rule. You cannot specify this clause when you are updating a shipment mapping rule. If you specify the DESCRIPTION clause when updating a shipment mapping rule, a caution message is issued and the DESCRIPTION clause is ignored.

HOST DATASET dataset-name MAPS TO REMOTE DATASET dataset-name

The HOST DATASET clause identifies the 1- to 44-character name or mask of the host data set name and the 1- to 44-character name or mask of the remote data set name. If you do not specify the MAPS TO REMOTE DATASET clause, the EXCLUDE clause is the default mapping rule. You must use the HOST DATASET clause when creating a shipment mapping rule. The clause is optional when updating a shipment mapping rule.

APPROXimate HOSt MEMBERS PER CYLinder nnn

Use this field to enter an approximation of the number of members that one cylinder might contain for data sets that map to this rule. You can enter in increments of .01 for values from .01 thru .99 and increments of 1 for values from 1 thru 999. If you omit this field or enter a 0, the field defaults to 16.

When packages are staged for shipment, the actual number of members being staged is divided by this value and the result is allocated plus 1 cylinder of primary and secondary space for the staging data sets. If you normally ship large members, you want this value to be a small number.

EXCLUDE

Use the EXCLUDE clause if you do not want to transmit the package outputs of a data set. You can only use the EXCLUDE clause if you do not specify the MAPS TO REMOTE DATASET clause. The two clauses are mutually exclusive.

Example: Define Shipment Mapping Rule SCL

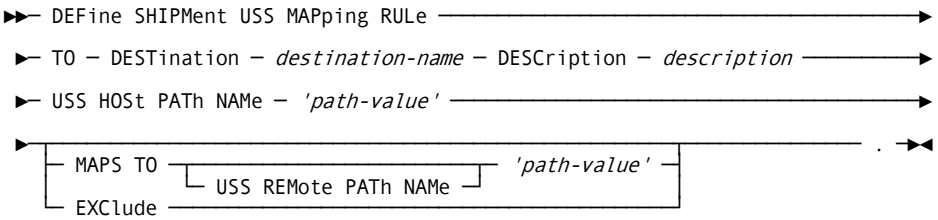
The following is an example of the DEFINE SHIPMENT MAPPING RULE SCL. The example creates a shipment rule for the shipment destination named BOSTNDM.

```
DEFINE SHIPMENT MAPPING RULE
          TO DESTINATION 'BOSTNDM'
          DESCRIPTION "MOVE OBJECTS"
          HOST DATASET 'ENDEVOR.QAFIN.OBJLIB*'
        MAPS TO REMOTE DATASET 'ENDEVOR.RMTFIN.OBJLIB*' .
```

The Define Shipment USS Mapping Rule Action

Use the DEFINE SHIPMENT USS MAPPING RULE action to create or update mapping rules between a host USS path name and a remote USS path name.

Define Shipment USS Mapping Rule Syntax



DEFINE SHIPMENT USS MAPPING RULE

Creates or updates a mapping rule between a host USS path name and a remote USS path name.

TO DESTINATION *destination-name*

Specifies the one- to seven-character name of an existing package shipment destination. Name maskName-maskName-masking is not supported.

DESCRIPTION *description*

Describes the USS mapping rule that is being created or updated. The text can be up to 40 characters. Embedded spaces are valid if the text is enclosed in either single or double quotation marks. This clause must be specified when creating a USS shipment mapping rule and is optional on an update.

USS HOST PATH NAME *path-value* MAPS TO USS REMOTE PATH NAME *path-value*

Specifies the 1- to 768-character host path name and the 1- to 768-character remote path name. Name-masking is supported. If the remote path name is omitted, the EXCLUDE clause is the default mapping rule. The HOST PATH NAME must be specified when creating or updating a mapping rule.

Valid path-values meet the following requirements:

- Must start and end with a forward slash (/). If the starting character is not a slash, a syntax error is issued. If the last character is not a slash, a slash is generated.
- If the path name contains a single quote character, use double quotes to enclose the text. A path name cannot contain both a single and a double quote character.
- If the path name cannot fit on a single statement, the overflow may be specified on subsequent lines. Enclose the first string in single quotes then follow it with a comma to indication that an additional line text exists. The pathname specification can not exceed 14 text string values.

If path-values are masked, then node masking resolutions between Host and Remote rules are resolved from right to left.

The MAPS TO clause can be specified with or without the USS REMote PATh NAMe sub-clause. Both of the following examples are valid:

```
MAPS TO path-value
MAPS TO USS REMote PATh NAMe path-value
```

On an update, if the MAPS TO clause is omitted, the EXCLUDE clause is assumed.

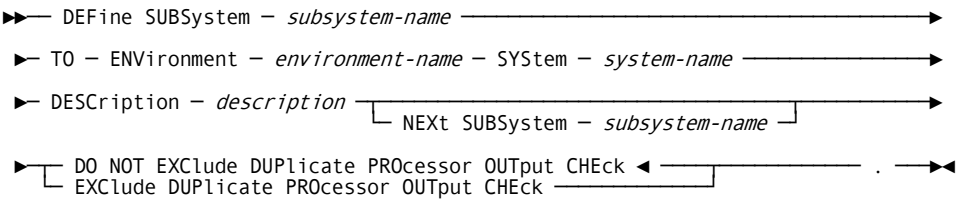
EXCLUDE

Does not transmit the USS outputs from the USS HOST PATH NAME specification. The EXCLUDE clause can only be used if the MAPS TO clause is omitted. The two clauses are mutually exclusive.

The Define Subsystem Action

Use the Define Subsystem statement to create or update a Subsystem definition.

Define Subsystem Syntax



DEFINE SUBSYSTEM *subsystem-name*

Specifies the one- to eight-character name of the Subsystem you are creating or updating. Name-masking is allowed. A name-masked Subsystem name updates all matching Subsystem definitions.

TO ENVIRONMENT *environment name* SYSTEM *system-name*

Specifies the inventory location to which you are defining the Subsystem. The environment name and system name must be fully specified; name-masking is not allowed.

DESCRIPTION *description*

Describes this Subsystem with text of up to 50 characters in length. If the text contains embedded spaces, enclose it in single or double quotation marks. The Description clause is required when you create a Subsystem definition. The clause is optional, when you update a Subsystem definition.

NEXT SUBSYSTEM *subsystem-name*

Specifies the name of the Subsystem in the next Environment. If you do not specify the Next Subsystem clause, the value defaults to the name of the Subsystem you are creating or updating.

DO NOT EXCLUDE | EXCLUDE DUPLICATE PROCESSOR OUTPUT CHECK

Specifies whether to exclude this Subsystem from being checked, if the Duplicate Processor Output Type Check is active for the System where this Subsystem is located. The default is Do Not Exclude Duplicate Processor Output Check.

Example: Define Subsystem SCL

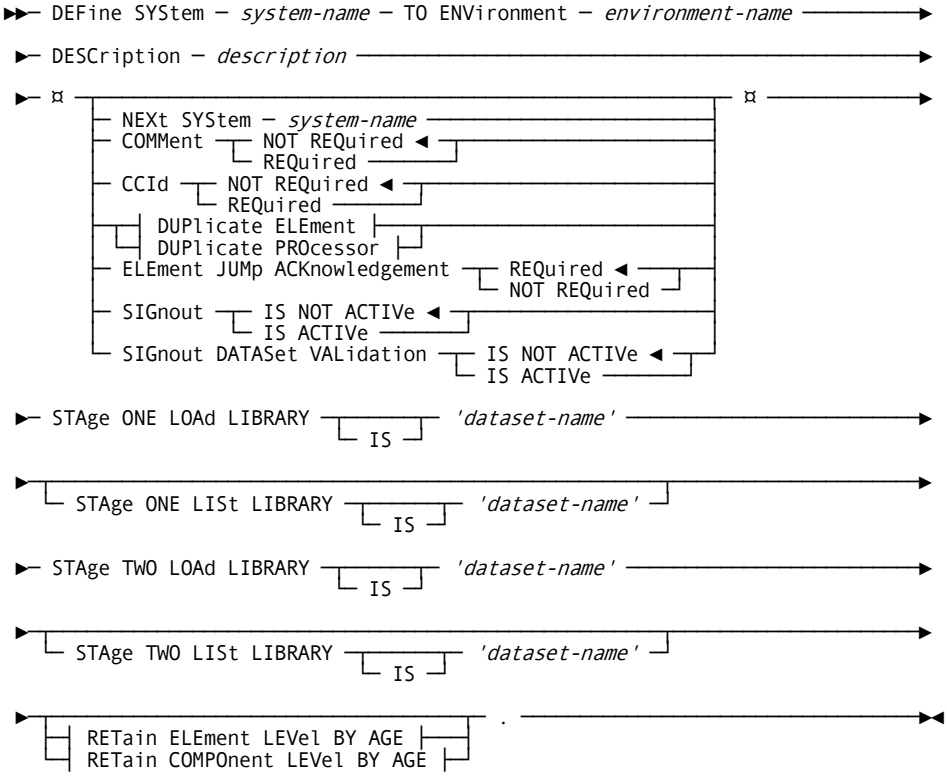
The following is an example of the Define Subsystem SCL. The example creates a Subsystem named GENLEDG for System ACCT.

```
DEFINE SUBSYSTEM "GENLEDG"
    TO ENVIRONMENT "DEVEL"
       SYSTEM "ACCT"
    DESCRIPTION "The General Ledger Subsystem"
    NEXT SUBSYSTEM "GENLEDG" .
```

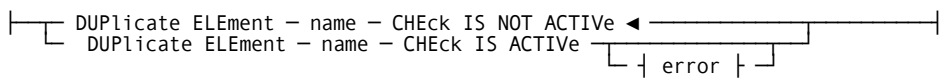
The Define System Action

Use the Define System statement to create or update System definitions.

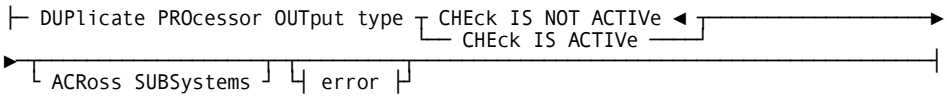
Define System Syntax



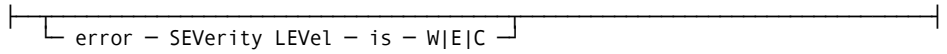
Expansion of DUPlicate ELEment



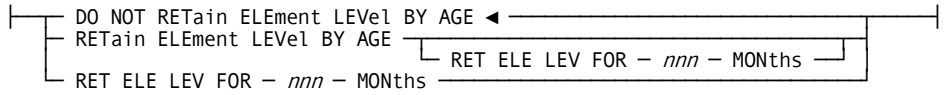
Expansion of DUPlicate PROcessor OUTput



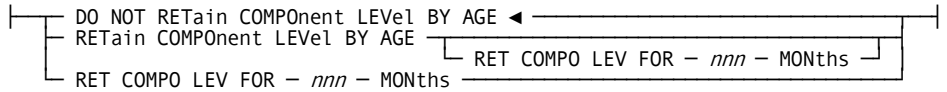
Expansion of error



Expansion of RETain ELEment LEVel BY AGE



Expansion of RETain COMPOnent LEVel BY AGE



The Define System statement includes the following clauses and options.

DEFINE SYSTEM *system-name*

Specifies the one- to eight-character name of the System you are creating or updating. Name maskName-masking is allowed. A name-masked System name updates all matching System definitions.

TO ENVIRONMENT *environment-name*

Specifies the Environment to which you are defining the System. The Environment name you specify must be fully qualified; name-masking is not allowed.

DESCRIPTION *description*

Describes this System with text of up to 50 characters in length. If the text contains embedded spaces, enclose it in single or double quotation marks. This clause is required when you create a System definition. The clause is optional, when you update a System definition.

NEXT SYSTEM *system-name*

Specifies the name of the System in the next Environment. If you do not specify this clause, the value defaults to the name of the System you are creating or updating.

COMMENTS REQUIRED | NOT REQUIRED

Specifies whether comments are required for actions against elements in this System. The default is COMMENTS NOT REQUIRED.

CCID REQUIRED | NOT REQUIRED

Specifies whether CCIDs are required for actions against elements in this System. The default is CCID *NOT REQUIRED*.

DUPLICATE ELEMENT NAME CHECK IS ACTIVE | IS NOT ACTIVE

Specifies whether the element name registration feature is active. If this clause is omitted, the default is DUPLICATE ELEMENT NAME CHECK IS *NOT ACTIVE*. The following option applies to this clause:

ERROR SEVERITY LEVEL

Specifies the message severity level for messages that result when errors are found during an element name registration check. This clause is only valid if the Define System statement includes the clause DUPLICATE ELEMENT NAME CHECK *IS ACTIVE*. Valid values are:

- **W** - Warning
- **C** - Caution
- **E** - Error - The default, if this clause is omitted.

DUPLICATE PROCESSOR OUTPUT TYPE CHECK IS ACTIVE | IS NOT ACTIVE

Specifies whether the duplicate processor output type check is active. If active, CA Endevor SCM compares element names across types and processors groups for the same processor output type. A conflict occurs if a user attempts to add or create an element and an element with the same name and processor output Type exists in the same System, with a different Type. If this clause is omitted, the default is DUPLICATE PROCESSOR OUTPUT TYPE CHECK IS *NOT ACTIVE*. The following options apply to this clause:

ACROSS SUBSYSTEMS

Specifies whether the check extends to the Subsystems defined to this System. This clause is only valid if DUPLICATE PROCESSOR OUTPUT TYPE CHECK *IS ACTIVE*.

ERROR SEVERITY LEVEL

Specifies the message severity level for messages that result when errors are found during a duplicate processor output Type check. This clause is only valid if the Define System statement includes the clause DUPLICATE PROCESSOR OUTPUT TYPE CHECK *IS ACTIVE*. Valid values are:

- **W** - Warning
- **C** - Caution
- **E** - Error - The default, if this clause is omitted.

ELEMENT JUMP ACKNOWLEDGMENT REQUIRED | NOT REQUIRED

Specifies whether users must use ACKNOWLEDGE ELM JUMP=Y when moving elements. CA Endevor SCM uses this field when it finds an element being moved at a nonmapped Stage between the from and to locations of the move. The default is ELEMENT JUMP ACKNOWLEDGMENT *REQUIRED*.

SIGNOUT IS ACTIVE | NOT ACTIVE

Specifies whether the signin/signout facility is active for this System. If this option is active, CA Endevor SCM performs signin and signout processing. The default is SIGNOUT IS *NOT ACTIVE*.

SIGNOUT DATASET VALIDATION IS ACTIVE | NOT ACTIVE

Specifies whether data set validation is active for this System. If this option is active, when an element is retrieved, the name of the data set to which the element was retrieved to (and member name if the data set is a library) is placed in the element master record. When an Add or Update action is performed against an element, CA Endevor SCM compares the *retrieve to* data set name (and member name if applicable) to the source input data set and member name specified for the action. If the data set (and member) names match, the action continues. If the names do not match, CA Endevor SCM checks the override signout value on the request. If the override signout value is Y, processing continues, if N, the action fails.

The default is SIGNOUT DATASET VALIDATION IS *NOT ACTIVE*.

STAGE ONE | TWO LOAD LIBRARY is *dataset-name*

Specifies the data set names of the processor load libraries for the System. The data set name must meet the following requirements:

- Fully qualified and no longer than 44 characters in length.
- Predefined and cataloged in the system catalog.
- Partitioned data set.
- Undefined record format (DCB=RECFM=U).

This clause is required when creating the System definition. The clause is optional when updating a System definition.

STAGE ONE | TWO LIST LIBRARY is *dataset-name*

Specifies the processor listing libraries for this system.

RETAIN ELEMENT LEVEL BY AGE | DO NOT RETAIN ELEMENT LEVEL BY AGE | RETAIN ELEMENT LEVEL FOR *nnn* MONTHS

Optional. Specifies whether old element delta levels are retained and for how many months they are retained.

When creating a system, the clause has the following effect:

- If you do not code any of the options, then the default is DO NOT RETAIN ELEMENT LEVEL BY AGE.
- You can code RETAIN ELEMENT LEVEL BY AGE alone or with RETAIN ELEMENT LEVEL FOR *nnn* MONTHS.
  - If you code RETAIN ELEMENT LEVEL BY AGE alone, then the retention time defaults to 999 months.
  - If you code RETAIN ELEMENT LEVEL FOR *nnn* MONTHS alone, then RETAIN ELEMENT LEVEL BY AGE is assumed.

When updating a system, the clause has the following effect:

- If the existing system is coded with DO NOT RETAIN ELEMENT LEVEL BY AGE and you want to enable the retention feature, then you must code RETAIN ELEMENT LEVEL BY AGE. If you want to specify a retention time less than 999 months, you must also code RETAIN ELEMENT LEVEL FOR *nnn* MONTHS.
- If the existing system is coded with RETAIN ELEMENT LEVEL BY AGE and you want to specify a retention time less than 999 months, then you only need to code RETAIN ELEMENT LEVEL FOR *nnn* MONTHS.
- If the existing system is coded with RETAIN ELEMENT LEVEL BY AGE and you want to turn off the retention feature, you must code DO NOT RETAIN ELEMENT LEVEL BY AGE.

RETAIN ELEMENT LEVEL BY AGE

Specifies that aged delta level retention for elements is in effect. If this clause is specified, but the RETAIN ELEMENT LEVEL FOR *nnn* MONTHS clause is not, then elements are retained for 999 months.

DO NOT RETAIN ELEMENT BY AGE

Specifies that aged delta level retention of element is *not* in effect. This clause is the default if neither RETAIN ELEMENT LEVEL BY AGE nor RETAIN ELEMENT LEVEL FOR *nnn* MONTHS are specified.

RETAIN ELEMENT LEVEL FOR *nnn* MONTHS

Specifies that aged delta level retention for elements is in effect and the number of months that element levels are retained. Valid values are 1 through 999. This clause can be specified alone or with RETAIN ELEMENT LEVEL BY AGE. When used either way the effect is the same.

RETAIN COMPONENT LEVEL BY AGE | DO NOT RETAIN COMPONENT LEVEL BY AGE | RETAIN COMPONENT LEVEL FOR *nnn* MONTHS

Optional. Specifies whether old component delta levels are retained and for how many months they are retained.

When creating a System, this clause has the following effects:

- If you do not code any of the options, then the default is DO NOT RETAIN COMPONENT LEVEL BY AGE.
- You can code RETAIN COMPONENT LEVEL BY AGE alone or with RETAIN COMPONENT LEVEL FOR *nnn* MONTHS.
  - If you code RETAIN COMPONENT LEVEL BY AGE alone, then the retention time defaults to 999 months.
  - If you code RETAIN COMPONENT LEVEL FOR *nnn* MONTHS alone, then RETAIN COMPONENT LEVEL BY AGE is assumed

When updating a system, this clause has the following effects:

- If the existing system is coded with DO NOT RETAIN COMPONENT LEVEL BY AGE and you want to enable the feature, then you must code RETAIN COMPONENT LEVEL BY AGE. If you want to specify a retention time less than 999 months, you must also code RETAIN COMPONENT LEVEL FOR *nnn* MONTHS.
- If the existing system is coded with RETAIN COMPONENT LEVEL BY AGE and you want to specify a retention time less than 999 months, then you only need to code RETAIN COMPONENT LEVEL FOR *nnn* MONTHS.
- If the existing system is coded with RETAIN COMPONENT LEVEL BY AGE and you want to turn off the feature, you must code DO NOT RETAIN COMPONENT LEVEL BY AGE.

RETAIN COMPONENT LEVEL BY AGE

Specifies that aged delta level retention for components is in effect. If RETAIN COMPONENT LEVEL BY AGE is specified, but the RETAIN COMPONENT LEVEL FOR *nnn* MONTHS clause is not, then components are retained for 999 months.

DO NOT RETAIN COMPONENT BY AGE

Specifies that aged delta level retention of component is *not* in effect. This clause is the default if neither RETAIN COMPONENT LEVEL BY AGE nor RETAIN COMPONENT LEVEL FOR *nnn* MONTHS are specified.

RETAIN COMPONENT LEVEL FOR *nnn* MONTHS

Specifies that aged delta level retention for components is in effect and the number of months that component levels are retained. Valid values are 1 through 999. This clause can be specified alone or with RETAIN COMPONENT LEVEL BY AGE. When used either way the effect is the same.

Example: Define System SCL

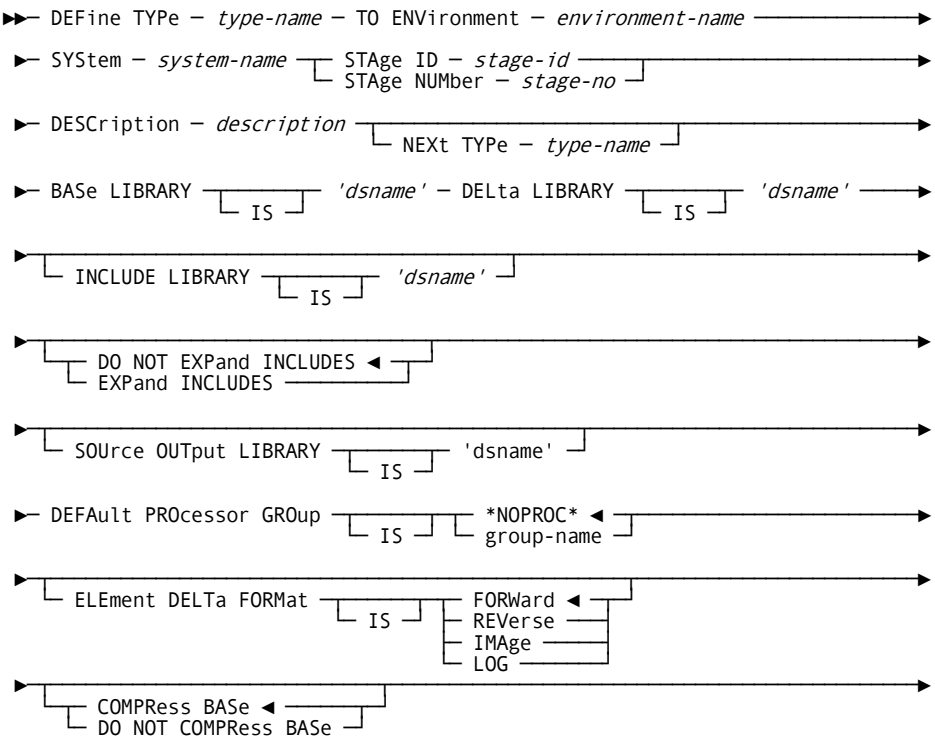
In this example DEFINE SYSTEM statement, a System named ACCT is created in the DEVEL environment. When a user performs an action against any element in this System, the user must enter a comment. In addition, CA Endevor SCM signin/signout processing and signout data set validation are active for this System. This statement also specifies the location of the optional Stage One and Stage Two List libraries.

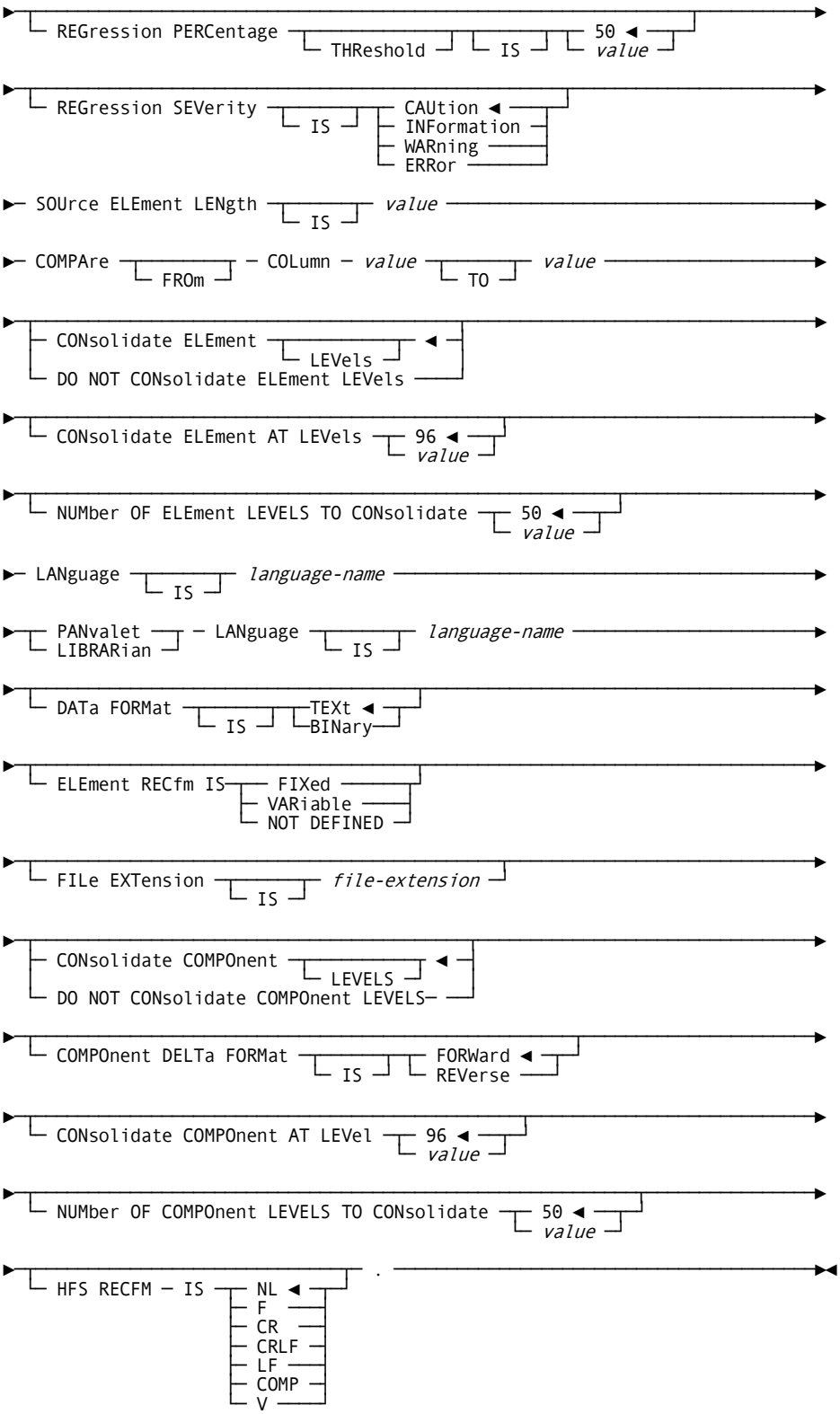
```
DEFINE SYSTEM "ACCT"
        TO ENVIRONMENT "DEVEL"
        DESCRIPTION "The Accounting System"
        COMMENTS REQUIRED
        SIGNOUT DATASET VALIDATION IS ACTIVE
        SIGNOUT IS ACTIVE
        STAGE ONE LOAD LIBRARY IS "ENDEVOR.LOAD1.ACCT"
        STAGE ONE LIST LIBRARY IS "ENDEVOR.LIST1.ACCT"
        STAGE TWO LOAD LIBRARY IS "ENDEVOR.LOAD2.ACCT"
        STAGE TWO LIST LIBRARY IS "ENDEVOR.LIST2.ACCT" .
```

The Define Type Action

Use the DEFINE TYPE action to create or update type definitions.

Define Type Syntax




REGression PERCentage
THReshold
IS
50 ◄
value
REGression SEVerity
IS
CAUtion ◄
INFormation
WARning
ERRor
SOUrce ELEment LENgth
IS
value
COMPAre
FROm
COLumn
value
TO
value
CONsolidate ELEment
LEVels
◄
DO NOT CONsolidate ELEment LEVels
CONsolidate ELEment AT LEVels
96 ◄
value
NUMber OF ELEment LEVELS TO CONsolidate
50 ◄
value
LANguage
IS
language-name
PANvalet
LIBRARian
LANguage
IS
language-name
DATa FORMat
IS
TEXt ◄
BINary
ELEment RECfm IS
FIXed
VARiable
NOT DEFINED
FILe EXTension
IS
file-extension
CONsolidate COMPOnent
LEVELS
◄
DO NOT CONsolidate COMPOnent LEVELS
COMPOnent DELTa FORMat
IS
FORWard ◄
REVerse
CONsolidate COMPOnent AT LEVel
96 ◄
value
NUMber OF COMPOnent LEVELS TO CONsolidate
50 ◄
value
HFS RECFM
IS
NL ◄
F
CR
CRLF
LF
COMP
V
.

DEFINE TYPE type-name

The DEFINE TYPE clause identifies the 1- to 8-character name of the type you are defining or updating. You can partially or fully wildcard the type name. A wildcarded type name updates all matching type definitions.

If you specify PROCESS as the type name you must specify the language name in the LANGUAGE clause as CNTLPROC. If you use the PANVALET/LIBRARIAN LANGUAGE clause, the language name you specify cannot be CNTLPROC. Likewise, if the type name you specify in the DEFINE TYPE clause is not PROCESS, then the language name you specify in the LANGUAGE clause and the PANVALET/LIBRARIAN cannot be CNTLPROC.

TO **ENVIRONMENT environment-name**

SYSTEM system-name

STAGE ID stage-id

STAGE NUMBER stage-no

The TO clause identifies the inventory location to which you are defining or updating the type. Names you specify in the TO clause must be fully specified and non-wildcarded. Enter the system name and either a stage ID or stage number to which the type is defined.

DESCRIPTION description

Use the DESCRIPTION clause to enter text of up to 50 characters in length describing this type. If the text contains imbedded spaces enclose it in either single or double quotation marks. You must use the DESCRIPTION clause when creating a type definition. The clause is optional when updating a type definition.

NEXT TYPE type-name

The NEXT TYPE clause identifies the name of the type at the next map location. If you do not specify the NEXT TYPE clause, it defaults to the name of the type you are defining.

BASE LIBRARY IS dataset-name

The BASE LIBRARY clause identifies the base library for this type. You must use a fully qualified data set name of no longer than 44 characters in length. The data set can include the &C1 symbolic variables. If the data set name does not contain a &C1 symbolic, the data set must be catalogued. If the data set name does not contain any &C1 symbolic values and the data set organization is partitioned, the data set record length must be greater than or equal to the SOURCE ELEMENT LENGTH. You must specify the BASE LIBRARY clause when creating a new type definition. The clause is optional when updating a type definition.

Note: For more information about using symbolics, see Rules for Using Symbolics in Dataset Names in Type Definition Parameters (see page 302).

DELTA LIBRARY IS dataset-name

The DELTA LIBRARY clause identifies the name of the delta library for the type. You must use a fully qualified data set name of no longer than 44 characters in length. The data set can include the &C1 symbolic variables. If the data set name does not contain a &C1 symbolic, the data set must be cataloged. If the data set name does not contain any &C1 symbolic values and the data set organization is partitioned, the data set record length must be greater than or equal to the SOURCE ELEMENT LENGTH. You must specify the DELTA LIBRARY clause when creating a new type definition. The clause is optional when updating a type definition.

Note: For more information about using symbolics, see Rules for Using Symbolics in Dataset Names in Type Definition Parameters (see page 302).

INCLUDE LIBRARY IS dataset-name

The INCLUDE LIBRARY clause identifies the name of the partitioned data set, CA Panvalet, CA Librarian, or ELIB containing the include members for the type. You must use a fully qualified data set name of no longer than 44 characters in length. If the data set name does not contain a &C1 symbolic, the data set must be catalogued. If the data set name does not contain any &C1 symbolic values and the data set organization is partitioned, the data set record length must be greater than or equal to the SOURCE ELEMENT LENGTH.

EXPAND/DO NOT EXPAND INCLUDES

You can specify if members should be expanded from this library. The default is DO NOT EXPAND INCLUDES. You can only specify the EXPAND INCLUDES clause if you specify the INCLUDE LIBRARY clause.

Note: For more information about using symbolics, see Rules for Using Symbolics in Dataset Names in Type Definition Parameters (see page 302).

SOURCE OUTPUT LIBRARY IS dataset-name

The SOURCE OUTPUT LIBRARY clause identifies the data set name of the source output library. If the data set name does not contain a &C1 symbolic, the data set must be cataloged. If the data set name contains any &C1 symbolic values and the data set organization is partitioned, the data set record length must be greater than or equal to the SOURCE ELEMENT LENGTH.

Note: For more information about using symbolics, see Rules for Using Symbolics in Dataset Names in Type Definition Parameters (see page 302).

DEFAULT PROCESSOR GROUP IS *NOPROC* | group-name

The DEFAULT PROCESSOR GROUP clause identifies the processor group for this type. The processor group name you use must be fully specified and not contain imbedded blanks. The default is *NOPROC*.

If you are creating a type definition and you do not specify the DEFAULT PROCESSOR GROUP clause and the type name you specify on the DEFINE TYPE action is 'PROCESS', the default value is set to PROCESS. If you are creating a type definition and you do not specify the DEFAULT PROCESSOR GROUP clause and the type name you specify in the DEFINE TYPE action is not 'PROCESS', the default value is set to *NOPROC*.

ELEMENT DELTA FORMAT IS FORWARD | REVERSE | IMAGE | LOG

The ELEMENT DELTA FORMAT clause specifies the delta storage format for elements of this type. If you do not specify the ELEMENT DELTA FORMAT clause and you are creating a type definition, the delta format defaults to FORWARD.

If sourced elements exist, you cannot change the element type definition *to log* delta or *to image* delta format from another format, nor can you change the definition *from log or from image* to another format.

COMPRESS | DO NOT COMPRESS BASE

The COMPRESS BASE clause indicates whether to compress the base form of elements stored in reverse delta format. The default is COMPRESS BASE.

REGRESSION PERCENTAGE THRESHOLD IS 50 | value

The REGRESSION PERCENTAGE clause specifies the maximum acceptable regression percent for elements of this type. You must specify a numeric value between 0 and 99, inclusive. If you are creating a type definition and do not specify this clause, the value defaults to 50. The use of a regression percentage of 0 turns off regression testing for that type — no change or base regression messages are issued. If the delta storage format for this type is full-image delta or log delta format, the regression percent must be 0.

You can abbreviate PERCENTAGE to PERC, but not to PER.

REGRESSION SEVERITY IS INFORMATION | WARNING | CAUTION | ERROR

The REGRESSION SEVERITY IS clause identifies the severity of the error message that issues when CA Endevor SCM detects regression. The default is CAUTION.

SOURCE ELEMENT LENGTH IS value

The SOURCE ELEMENT LENGTH identifies the logical record length in source statements. The maximum allowable value is 32,000. If the type is PROCESS, the source element length must be exactly 80. You must specify the SOURCE ELEMENT LENGTH clause when creating a type definition. The clause is optional when updating a type definition. When updating an existing type, do not modify this value if any elements of that type already exist.

COMPARE FROM COLUMN value TO value

The COMPARE FROM COLUMN clause identifies the position within each statement at which CA Endevor SCM begins comparing to identify changed statements. The values you specify must be between one and the SOURCE ELEMENT LENGTH. The value you specify in the COMPARE FROM clause must be less than or equal to the value you specify in the COMPARE TO clause. You must specify the COMPARE FROM COLUMN clause when creating a type definition. The clause is optional when updating a type definition.

CONSOLIDATE | DO NOT CONSOLIDATE ELEMENT LEVELS

The CONSOLIDATE ELEMENT LEVEL clause identifies whether or not CA Endevor SCM is to consolidate element change levels. The default is to consolidate change levels.

If you specify the DO NOT CONSOLIDATE ELEMENT LEVELS clause you must set the corresponding NUMBER OF ELEMENT LEVELS TO CONSOLIDATE clause to zero. If you are updating a type definition, and specify the DO NOT CONSOLIDATE ELEMENT LEVELS clause but do not specify the NUMBER OF ELEMENT LEVELS TO CONSOLIDATE clause, the NUMBER OF LEVELS TO CONSOLIDATE clause is set to zero and you receive a warning message.

If you specify the CONSOLIDATE ELEMENT LEVELS clause, the value you specify in the corresponding NUMBER OF ELEMENT LEVELS TO CONSOLIDATE clause must be greater than zero.

CONSOLIDATE ELEMENT AT LEVEL 96 | value

The CONSOLIDATE ELEMENT AT LEVEL clause specifies the level number at which CA Endevor SCM consolidates change levels. The default is 96.

NUMBER OF ELEMENT LEVELS TO CONSOLIDATE 50 | value

The NUMBER OF LEVELS TO CONSOLIDATE clause indicates the number of deltas to consolidate when the number of levels reaches the figure in the CONSOLIDATE ELEMENT AT LEVEL clause. You can specify a value between 1 and 96. The default is 50.

HFS RECFM

Identifies the record delimiter used in a HFS file. A record delimiter is necessary due to the nature of HFS files. HFS files contain one large data stream; therefore, a delimiter is used to identify individual records within that data stream. If a delimiter is not specified, the system defaults to NL.

Acceptable delimiter values are:

- **COMP** – Variable length records compressed by CA Endevor SCM
- **CR** – Carriage return. ASCII and EBCDIC value "CR". The hex value is '0D'.
- **CRLF** – EBCDIC carriage return \ line feed. The hex value is '0D25'.
- **CRNL** – ASCII carriage return \ new line. The hex value is '0D0A'.
- **F** – Fixed Length
- **LF** – EBCDIC line feed. The hex value is '25'.
- **NL** – Default. EBCDIC new line character. This is the delimiter is used by the OEDIT and OBROWS Eeditor.
- **V** – Variable. The first two bytes of the record contain the RDW (record descriptor word). The RDW contains the length of the entire record, including the RDW.

LANGUAGE IS language-name

The LANGUAGE clause defines the source language of the type. You can use any alphanumeric string of up to eight characters in length. You must specify the LANGUAGE clause when creating a type definition. The clause is optional when updating a type definition.

If the language you specify in the LANGUAGE IS clause is one of the languages in the following table, the values you specify in the COMPARE FROM clause are compared to determine if they fall within the following ranges.

Language	Compare From	Compare To
ANSCOBOL	7	72
ASM	1	72
ASSEMBLR	1	72
BAL	1	72
COBOL	7	72
COBOL-72	7	72
COBOL-74	7	72
COBOLF	7	72
COBOLVS	7	72

Language	Compare From	Compare To
DATA	1	80
FORTRAN	1	72
FORT	1	72
JCL	1	72
LNKCARD	1	72
PL1	1	72
PLI	1	72
PL/1	1	72
PL/I	1	72
RPG	6	74

PANVALET | LIBRARIAN LANGUAGE IS language-name

The PANVALET/LIBRARIAN LANGUAGE clause identifies the CA Panvalet or CA Librarian source language for this type. You must specify this clause when creating a type definition and if CA Panvalet or CA Librarian support is not active. The clause is optional when updating a type definition. If you do not specify the PANVALET/LIBRARIAN LANGUAGE clause but specify a LANGUAGE clause, CA Endevor SCM uses the language name in the LANGUAGE clause for the PANVALET/LIBRARIAN LANGUAGE.

The following table lists acceptable names for CA Panvalet as well as values within which the names must fall for the COMPARE FROM clause.

CA Panvalet Language	Compare From	Compare To
ALC	1	72
ANSCOBOL	7	72
AUTOCODE	6	75
BAL	1	72
COBOL	7	72
COBOL-72	7	72
DATA	1	80
FORTRAN	1	72
JCL	1	72
OBJECT	1	72

CA Panvalet Language	Compare From	Compare To
OTHER	1	72
PL/1	1	72
PL/I	1	72
RPG	6	74
USER180	1	80
USER780	7	80

The following table lists acceptable names for CA Librarian as well as values within which the names must fall for the COMPARE FROM clause.

CA Librarian Language	Compare From	Compare To
ASM	1	72
CBL	7	72
CB2	7	72
CB3	7	72
COB	7	72
DAT	1	80
DB2	1	72
FOR	1	72
JCL	1	72
PLI	1	72
RPG	6	74
SAS	1	80
TXT	1	80
VSB	1	80

DATa FORMat IS BINary | TEXt

The DATA FORMAT clause indicates whether the element is in binary or text format.

ELEment RECfm IS FIXed | VARiable | NOT DEFINED

Specifies whether the element has fixed length records or variable length records. This parameter is used by the CA Endevor Quick Edit option only. During a Quick Edit session, a temporary ISPF edit data set is allocated to hold the element to be edited. The edit data set is allocated either as variable or fixed. If this clause is omitted or Not Defined is specified, the record RECFM is determined by the input data set record lengths. For example, the data set is allocated as variable if the element already exists and the element records have varying lengths. It is allocated as fixed if the element already exists and all the element records are the same length. It is always allocated as fixed if a Quick Edit Create command is executed to create a new element. If it is allocated as fixed length, the record length is the Type source length defined by the Source Element Length parameter.

FILE EXTENTION IS file-extension

The FILE EXTENTION clause identifies the eight-character file extension for this element. Valid values are a-z, A-Z, 0-9, or all blanks. Trailing blanks are supported, but embedded blanks are not.

Note: For more information about data format and file extensions, see the *CA CM Enterprise Workbench Installation and Configuration Guide*.

CONSOLIDATE | DO NOT CONSOLIDATE COMPONENT LEVELS

The CONSOLIDATE COMPONENT LEVEL clause identifies whether or not CA Endevor SCM is to consolidate component change levels. The default is to consolidate change levels.

If you specify the DO NOT CONSOLIDATE COMPONENT LEVELS clause you must set the corresponding NUMBER OF COMPONENT LEVELS TO CONSOLIDATE clause to zero. If you are updating a type definition and specify the DO NOT CONSOLIDATE COMPONENT LEVELS clause but do not specify the NUMBER OF COMPONENT LEVELS TO CONSOLIDATE clause, the NUMBER OF COMPONENTS LEVELS TO CONSOLIDATE clause is set to zero and you receive a warning message.

If you specify the CONSOLIDATE COMPONENT LEVELS clause, the value you specify in the corresponding NUMBER OF COMPONENTS LEVELS TO CONSOLIDATE clause must be greater than zero.

COMPONENT DELTA FORMAT IS FORWARD | REVERSE

The COMPONENT DELTA FORMAT clause specifies the delta storage format for component list information. If you do not specify this clause and you are creating a type definition, the delta format defaults to FORWARD.

CONSOLIDATE COMPONENT AT LEVEL 96 | value

The CONSOLIDATE COMPONENT AT LEVEL clause specifies the level number at which CA Endevor SCM consolidates change levels. You can specify a value between 1 and 96. The default is 96.

NUMBER OF COMPONENT LEVELS TO CONSOLIDATE 50 | value

The NUMBER OF COMPONENT LEVELS TO CONSOLIDATE clause indicates the number of deltas to consolidate when the number of levels reaches the figure in the CONSOLIDATE COMPONENTS AT LEVEL clause. You can specify a value between 1 and 96. The default is 50.

Example: Define Type SCL

The following is an example of the DEFINE TYPE SCL. The example creates a type named COBOL.

```
DEFINE  TYPE "COBOL"
   TO   ENVIRONMENT "DEVEL"
                SYSTEM "ACCT"
                STAGE NUMBER 1
   DESCRIPTION "The TYPE COBOL"
   BASE LIBRARY IS "ENDEVOR.BASE1.SOURCE"
        DELTA LIBRARY IS "ENDEVOR.DELTA1.SOURCE"
    INCLUDE LIBRARY IS "ENDEVOR.INCLLIB.COPYBOOK"
   DEFAULT PROCESSOR GROUP IS "COBNBL1"
   ELEMENT DELTA FORMAT IS REVERSE
   SOURCE ELEMENT LENGTH IS 80
   COMPARE FROM COLUMN 7 TO 72
   CONSOLIDATE ELEMENT AT LEVEL 80
   NUMBER OF ELEMENT LEVELS TO CONSOLIDATE 30
   LANGUAGE IS "COBOL"
   PANVALET LANGUAGE IS "COBOL" .
```

Rules for Using Symbolics in Data Set Names in Type Definition Parameters

Endevor symbols (&C1) and user-symbols (&#) can be used in the data set definitions for the TYPE definition parameters BASE LIBRARY, DELTA LIBRARY, INCLUDE LIBRARY, and SOURCE OUTPUT LIBRARY. In addition, Endevor symbols and user-defined symbols can be nested to form a valid user-defined symbol name that is resolved at run time. A character string beginning with &# can also be concatenated with one or more Endevor symbols to form a valid user-defined symbol name that is resolved at run time.

Be careful when coding both nested and concatenated User defined and Endevor defined symbols. When using these coding techniques the data set name is not finally resolved until run time. If an incorrect data set name is specified the result could be an allocation error at run time.

Example--Endevor Symbol Defined in BASE LIBRARY IS dataset-name

This example shows an Endevor symbol (&C1) in the BASE LIBRARY IS dataset-name.

```
IPRFX.IQUAL.&C1SUBSYS.BASE
```

Example--User-defined Symbol in BASE LIBRARY IS dataset-name

This example shows a user-defined symbol (&#) defined in the dataset-name.

```
IPRFX.IQUAL.&#USERVAL.BASE
```

Example-Nested User-defined Symbol in BASE LIBRARY IS dataset-name

This example shows nested user-defined symbol that forms a valid user-defined symbol name that will be resolved at run time.

```
IPRFX.IQUAL.&#&SUBSYS.BASE
```

In this example &#&C1SUBSYS would be resolved at run time to &#PAY which would then be resolved to the value defined for the User symbol #PAY.

Example-Nested Character String Beginning with &# in BASE LIBRARY IS dataset-name

This example shows a character string beginning with &# can also be concatenated with one or more Endevor symbols to form a valid User defined symbol name that will be resolved at run time.

```
IPRFX.IQUAL.&#DIV&SUBSYS.BASE
```

In this example &#DIV&C1SUBSYS would be resolved at run time to &#DIVPAY which would then be resolved to the value defined for the User

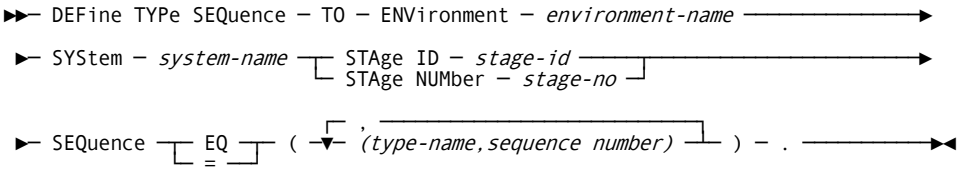
symbol #DIVPAY.

The Define Type Sequence Action

Use the DEFINE TYPE SEQUENCE action to update the relative sequence of processing for the various element types defined within a system. The DEFINE TYPE SEQUENCE action merges the existing type sequence information with the sequence information you specify in the DEFINE TYPE SEQUENCE action. If the option Global Type Sequencing is in effect at your site, a DEFINE TYPE SEQUENCE action generates a warning message; however the SCL statement is still executed.

Note: It is not possible to create a type sequence using the DEFINE TYPE SEQUENCE action. The type sequence definition is automatically created when you define a new system.

Define Type Sequence Syntax



DEFINE TYPE SEQUENCE

The DEFINE TYPE SEQUENCE clause indicates that you are updating the relative sequence of processing for the various element types defined within a system. You must specify this clause.

TO ENVIRONMENT *environment-name*
SYSTEM *system-name*
STAGE ID *stage-id*
STAGE NUMBER *stage-no*

The TO clause identifies the inventory location to which you are defining the type sequence. Names you specify on the TO clause must be fully qualified.

SEQUENCE EQ/= *type-name,sequence number*

The SEQUENCE clause defines the processing sequence for processors for the types you indicate. The SEQUENCE clause contains a list of type name and sequence number pairs. The type name represents an existing element type name that must be defined to the system and stage you specify. The sequence number is a number between 0 and 9999 that is not a multiple of 10. You can specify type names and sequence numbers more than once. You must specify at least one type name and sequence number pair. Enclose multiple pairs in parentheses and separate by commas. For example, to define the type sequence for types COBOL and COPYBOOK, the SEQUENCE clause would be specified as:

```
SEQUENCE=((COBOL,15),(COPYBOOK,25))
```

The DEFINE TYPE SEQUENCE action merges the existing type sequence information with the sequence information you specify in the DEFINE TYPE SEQUENCE action. The merged information is used to create the new type sequence record. The existing type sequence numbers are assigned 10 and are incremented by 10. For example, assume the type sequence record for system Payroll is defined as follows:

```
COPYBOOK, MACRO, COBOLPGM, ASSEMPGM, JCL.
```

The COPYBOOK type is assigned sequence number 10, MACRO is assigned 20, COBOLPGM is assigned 30 and so on. Further, assume that the DEFINE TYPE SEQUENCE action specified the following SEQUENCE clause:

```
SEQUENCE=((COBOLPGM,35),(ASSEMPGM,25)).
```

The updated TYPE SEQUENCE record would be defined as follows:

```
COPYBOOK, MACRO, ASSEMPGM, COBOLPGM,JCL
```

Example: Define Type Sequence SCL

The following is an example of the DEFINE TYPE SEQUENCE SCL. The example updates the type sequence for system ACCT.

```
DEFINE  TYPE SEQUENCE
    TO  ENVIRONMENT "DEVEL"
                SYSTEM "ACCT"
                STAGE ID "U"
                SEQUENCE = ( (LINK, 05),
                             (COBOL, 15),
                             (ASM, 25),
                             (JCL, 35) ) .
```

Before the update the sequence numbers are:

```
LINK = 5
COBOL = 15
ASM = 25
JCL = 35
```

After the update the sequence numbers assigned are:

```
LINK = 10
COBOL = 20
ASM = 30
JCL = 40
```

Delete Statements

Use DELETE statements to delete existing environment definitions. The DELETE action will not delete an environment definition if any elements are defined to the environment definition. For example, you cannot delete a system if any subsystems or types are defined to that system. You must delete the subsystems and types before you can delete the system.

Note: There is no action to delete a type sequence definition. The DELETE SYSTEM action deletes the type sequence definitions.

The following general conventions apply to all DELETE statements:

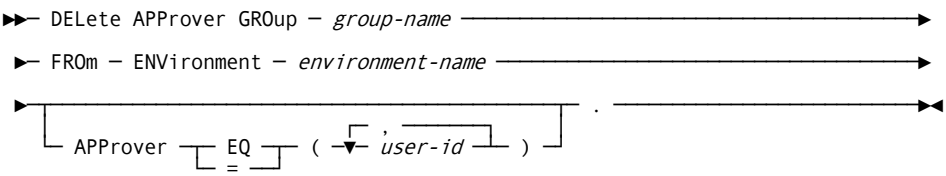
- Names you specify in the DELETE clause can have a maximum of 8 characters with the exception of SHIPMENT DESTINATION, which can be no longer than 7 characters, and APPROVER GROUP, which can be no longer than 16 characters.
- The DELETE **action** allows wildcarded names in all non-environment inventory location fields.

- Environment names you specify in the FROM clause must all be fully specified.
- Names cannot include embedded spaces, non-alphabetical, non-numeric, or non-national characters.

The Delete Approver Group Action

Use the DELETE APPROVER GROUP action to delete approver group definitions and any associated approver relate rules.

Delete Approver Group Syntax



DELETE APPROVER GROUP group-name

The DELETE APPROVER GROUP clause identifies the up to sixteen character name of the approver group you are deleting. You can use a partially or fully wildcarded approver group name.

FROM ENVIRONMENT environment-name

The FROM clause identifies the environment location of the approver group you are deleting. You must specify a fully qualified environment name.

APPROVER EQ/= user-id

The APPROVER clause identifies one or more approver user IDs to be deleted from the Approver Group definition. If you specify more than one user ID, enclose the IDs in parentheses and separate by commas. If you do not specify the APPROVER clause, CA Endevor SCM deletes the entire Approver Group definition.

Example: Delete Approver Group SCL

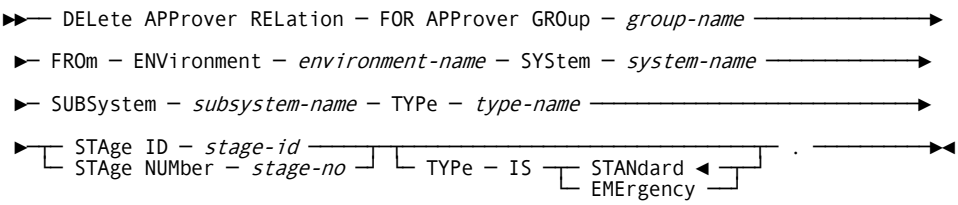
The following is an example of the DELETE APPROVER GROUP SCL. The example deletes certain approvers from the approver group named ACCTPAY1. Using the DELETE statement in this way acts like an update. The list of users that are currently in approver group ACCTPAY1 are (USER001, USER002, USER003, USER004, USER005). This example removes USER001 and USER004 from the Approver Group.

```
DELETE APPROVER GROUP "ACCTPAY1"
          FROM ENVIRONMENT "DEVEL"
          APPROVER EQ ( USER001,
                        USER004 ).
```

The Delete Approver Relation Action

Use the DELETE APPROVER RELATION action to delete an approver relation definition.

Delete Approver Relation Syntax



DELETE APPROVER RELATION

The DELETE APPROVER RELATION clause indicates that you are deleting a approver relation definition. You must specify this clause.

FOR APPROVER GROUP group-name

Specifies the name of the approver group associated with the approver relationship. You can specify a name-masked approver group name. If the value is name-masked, the value expands during processing and all matching references are deleted.

FROM ENVIRONMENT environment-name
SYSTEM system-name
SUBSYSTEM subsystem-name
TYPE type-name
STAGE ID stage-id
STAGE NUMBER stage-no

The FROM clause identifies the inventory location to which the approver group is related. You must use a fully specified environment name. You can fully specify or fully wildcard the system name, subsystem name, and type name. They cannot be partially wildcarded. The stage number must be 1 or 2. The stage ID can be explicit or an asterisk (*) wildcard to indicate both stages.

TYPE IS STANDARD/EMERGENCY

The TYPE IS clause identifies the approver type for this approver group. An approver group designated as standard can only approve standard packages. Likewise, an approver group designated as emergency can only approve emergency packages. The default is TYPE IS STANDARD.

Example: Delete Approver Relation SCL

The following is an example of the DELETE APPROVER RELATION SCL. The example deletes an approver relation for approver group ACCTPAY1.

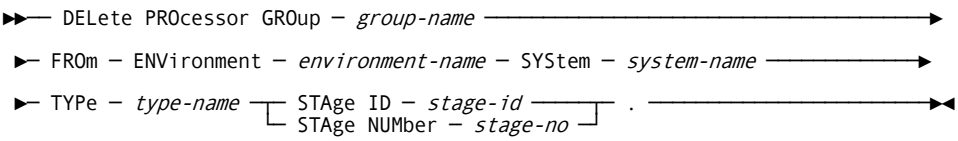
```
DELETE  APPROVER RELATION
   FOR  APPROVER GROUP "ACCTPAY1"
   FROM ENVIRONMENT "DEVEL"
                SYSTEM "ACCT"
                SUBSYSTEM "GENLEDG"
                TYPE "COBOL"
                STAGE NUMBER 1
                TYPE IS STANDARD .
```

The Delete Processor Group Action

Use the DELETE PROCESSOR GROUP action to delete processor group definitions. The DELETE PROCESSOR GROUP action also deletes all processor group symbols associated with the processor group.

Note: You cannot delete a processor group if the processor group is associated with an element at that stage.

Delete Processor Group Syntax



DELETE PROCESSOR GROUP group-name

The DELETE PROCESSOR GROUP clause identifies the up to eight character name of the processor group you are deleting. You can partially or fully wildcard the processor group name.

FROM ENVIRONMENT environment-name
SYSTEM system-name
TYPE type-name
STAGE ID stage-id
STAGE NUMBER stage-no

The FROM clause identifies the inventory location to which the processor group you are deleting is defined. Names you use in the FROM clause must be fully specified.

Example: Delete Processor Group SCL

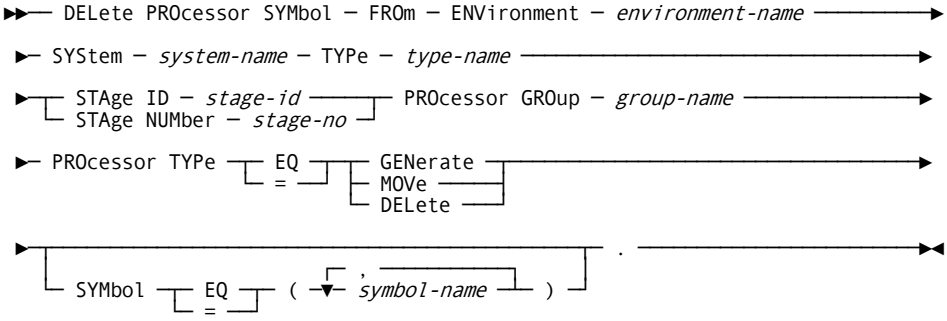
The following is an example of the DELETE PROCESSOR GROUP SCL. The example deletes all processor groups in environment DEVEL, system ACCT, type COBOL, and stage ID "U".

```
DELETE  PROCESSOR GROUP "*"
   FROM ENVIRONMENT "DEVEL"
        SYSTEM "ACCT"
                TYPE "COBOL"
                STAGE ID "U".
```

The Delete Processor Symbol Action

Use the DELETE PROCESSOR SYMBOL action to delete processor symbol overrides.

Delete Processor Symbol Syntax



DELETE PROCESSOR SYMBOL

The DELETE PROCESSOR SYMBOL clause indicates that you are deleting symbols in processors. You must specify this clause.

FROM ENVIRONMENT environment-name
SYSTEM system-name
TYPE type-name
STAGE ID stage-id
STAGE NUMBER stage-no
PROCESSOR GROUP group-name
PROCESSOR TYPE EQ/=GENERATE/MOVE/DELETE

The FROM clause identifies the inventory location of the processor group to which the processor symbols are defined, the processor group name, and a processor type within the group. The environment name, system name, and type name you specify in the FROM clause must all be fully specified.

You can fully specify, partially wildcard or fully wildcard the processor group name. The processor group must exist in the processor load library. The processor group name cannot be '*NOPROC*. Specify either a generate, move, or delete processor type for this processor symbol.

SYMBOL EQ/= symbol-name

The SYMBOL clause identifies one or more symbol names that are to be deleted from the symbol override. If you specify more than one name, enclose the symbol names in parentheses and separate by commas. Deleted symbols revert to the defaults specified in the processor definition. If you omit the SYMBOL clause, all the symbols associated with the processor group are deleted.

Examples: Delete Processor Symbol SCL

The following example deletes all generate processor symbols from processor group COBNBL1 in environment DEVEL, system ACCT, and stage ID U:

```
DELETE  PROCESSOR SYMBOL
   FROM ENVIRONMENT "DEVEL"
        SYSTEM "ACCT"
        TYPE "COBOL"
        STAGE ID "U"
        PROCESSOR GROUP "COBNBL1"
        PROCESSOR TYPE GENERATE.
```

This example deletes generate processor symbols SYMBOL1 and SYMBOL2 from processor group COBNBL1 in environment DEVEL, system ACCT, and stage ID U:

```
DELETE  PROCESSOR SYMBOL
   FROM ENVIRONMENT "DEVEL"
        SYSTEM "ACCT"
        TYPE "COBOL"
        STAGE ID "U"
        PROCESSOR GROUP "COBNBL1"
        PROCESSOR TYPE GENERATE
   SYMBOL EQ SYMBOL1
   SYMBOL EQ SYMBOL2.
```

This example also deletes generate processor symbols SYMBOL1 and SYMBOL2 from processor group COBNBL1 in environment DEVEL, system ACCT, and stage ID U:

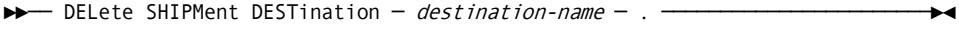
```
DELETE  PROCESSOR SYMBOL
   FROM ENVIRONMENT "DEVEL"
        SYSTEM "ACCT"
        TYPE "COBOL"
        STAGE ID "U"
        PROCESSOR GROUP "COBNBL1"
        PROCESSOR TYPE GENERATE
   SYMBOL EQ (SYMBOL1,SYMBOL2)
```

The Delete Shipment Destination Action

Use the DELETE SHIPMENT DESTINATION clause to delete destinations to which you ship package outputs.

Note: The DELETE SHIPMENT DESTINATION clause also deletes all the mapping rules that are associated with the destination. Data set mapping rules and USS mapping rules are deleted.

Delete Shipment Destination Syntax



DELETE SHIPMENT DESTINATION destination-name

The DELETE SHIPMENT DESTINATION clause identifies the up to seven-character name of the destination you are deleting. You can partially or fully wildcard the destination name.

Example: Delete Shipment Destination SCL

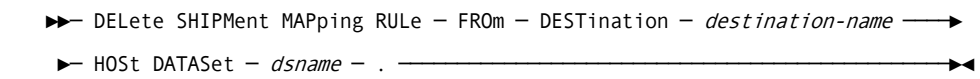
An example of the DELETE SHIPMENT DESTINATION SCL statement follows. The example deletes the shipment destination named "BOSTNDM".

```
DELETE SHIPMENT DESTINATION "BOSTNDM" .
```

The Delete Shipment Mapping Rule Action

Use the DELETE SHIPMENT MAPPING RULE action to delete mapping rules between a host data set name and a remote data set name.

Delete Shipment Mapping Rule Syntax



FROM DESTINATION destination-name

The FROM DESTINATION clause identifies the up to seven character name of an existing package shipment destination from which you are deleting a mapping rule. You must use a fully specified value.

HOST DATASET dsname

This clause identifies the up to 44 character name or mask of the host data set name.

Example: Delete Shipment Mapping Rule SCL

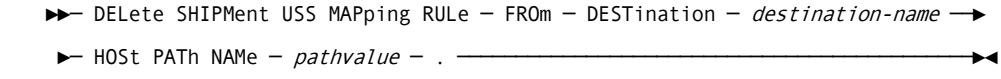
The following is an example of the DELETE SHIPMENT MAPPING RULE SCL. The example deletes a shipment mapping rule from shipment destination named "BOSTNDM".

```
DELETE SHIPMENT MAPPING RULE
   FROM SHIPMENT DESTINATION "BOSTNDM"
   HOST DATASET "ENDEVOR.QAFIN.SRCOUT" .
```

The Delete Shipment USS Mapping Rule Action

Use the DELETE SHIPMENT USS MAPPING RULE action to delete mapping rules between a USS host path name and a USS remote path name.

Delete Shipment USS Mapping Rule Syntax



FROM DESTINATION *destination-name*

Specifies the name of an existing package shipment destination from which you are deleting a mapping rule. Name-masking is *not* supported.

HOST PATH NAME *path-value*

Specifies the1- to 768-character host path name. Name-masking is supported. Valid path-values meet the following requirements:

- The path-value must start and end with a forward slash (/). If the starting character is not a slash, a syntax error is issued. If the last character is not a slash, a slash is generated.
- If the path name cannot fit on a single statement, the overflow can be specified on subsequent lines. Enclose the first string in single quotes then follow it with a comma to indication that an additional line text exists. The pathname specification cannot exceed 14 text string values.
- If the path name contains a single quote character, enclose it in double quotes.

For example, the following path-value in this clause is properly specified:

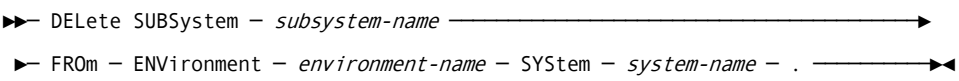
```
USS REMOTE PATH NAME PREFIX  '/u/users/endevor',
       '/stg/'
```

The Delete Subsystem Action

Use the DELETE SUBSYSTEM action to delete a subsystem definition. The DELETE SUBSYSTEM action also removes approver group junction definitions associated with the subsystem.

Note: You cannot delete a subsystem if any elements are associated with the subsystem at either Stage 1 or Stage 2.

Delete Subsystem Syntax



DELETE SUBSYSTEM subsystem-name

The DELETE SUBSYSTEM clause identifies the 1- to 8-character name of the subsystem you are deleting. You can partially or fully wildcard subsystem name.

FROM ENVIRONMENT environment-name
SYSTEM system-name

The FROM clause identifies the inventory location to which the subsystem is defined. Names you use in the FROM clause must be fully specified.

Example: Delete Subsystem SCL

The following is an example of the DELETE SUBSYSTEM SCL. The example deletes all subsystems using environment DEVEL, and system ACCT.

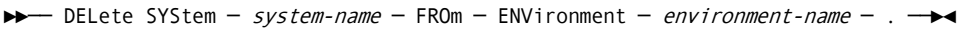
```
DELETE  SUBSYSTEM "*"
   FROM ENVIRONMENT "DEVEL"
                SYSTEM "ACCT" .
```

The Delete System Action

Use the DELETE SYSTEM action to delete a system definition. The DELETE SYSTEM action also deletes the type sequence definition at both Stage 1 and Stage 2.

Note: You cannot delete a system if any subsystems or types are associated with it at either Stage 1 or Stage 2.

Delete System Syntax



DELETE SYSTEM system-name

The DELETE SYSTEM clause identifies the 1- to 8-character name of the system you are deleting. You can partially or fully wildcard the system name.

FROM ENVIRONMENT environment-clause

The FROM ENVIRONMENT clause identifies the environment to which the system is defined. You must use fully specified environment name.

Example: Delete System SCL

The following is an example of the DELETE SYSTEM SCL. The example deletes a system named ACCT from environment DEVEL.

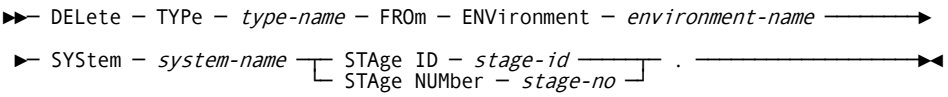
```
DELETE SYSTEM "ACCT"
   FROM ENVIRONMENT "DEVEL" .
```

The Delete Type Action

Use the DELETE TYPE action to delete a type definition. The DELETE TYPE action also removes any processor group and processor group symbol records associated with the element type.

Note: You cannot delete a type definition if there are any elements associated with it at the stage specified.

Delete Type Syntax



DELETE TYPE type-name

The DELETE TYPE clause indicates that you are deleting a type definition. You can specify a name-masked type name.

FROM ENVIRONMENT environment-name
SYSTEM system-name
STAGE ID stage-id
STAGE NUMBER stage-no

The FROM clause identifies the inventory location to which the type is defined. The environment and stage must be explicitly specified. You can specify a name-masked system name.

Example: Delete Type SCL

The following is an example of the DELETE TYPE SCL. The example deletes all types using environment DEVEL, system ACCT, and stage ID U.

```
DELETE  TYPE "*"
   FROM ENVIRONMENT "DEVEL"
                SYSTEM "ACCT"
                STAGE ID "U".
```