CA Endevor® Software Change Manager

Scenario 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 Endevor® Software Change Manager Parallel Development (CA Endevor Parallel Development)
- CA Endevor® Software Change Manager Quick Edit (CA Endevor Quick Edit)
- CA Endevor® Software Change Manager External Security Interface (CA Endevor External Security Interface)
- CA Endevor® Software Change Manager InfoMan Interface (CA Endevor InfoMan Interface)
- CA Endevor® Software Change Manager CA Librarian® Interface (CA Endevor CA Librarian Interface)
- CA Endevor® Software Change Manager Integration for the Natural Environment (CA Endevor Integration for the Natural Environment)
- CA Endevor® Software Change Manager for CA Panvalet® Interface (CA Endevor for CA Panvalet Interface)
- CA Endevor® Software Change Manager CA Roscoe® Interface (CA Endevor CA Roscoe Interface)
- CA Librarian® Base for z/OS (CA Librarian)
- CA Panvalet® for z/OS (CA Panvalet for z/OS)

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

How to Automatically Generate Using Element

- Concurrent Action Processing Overview (see page 28) and How Concurrent Action Processing Works (see page 29)— Updated to include the Alter action as an action that can be processed concurrently.

Package Ship Facility

- Mapping Rules (see page 102)— For the Package Ship facility, updated to specify that matching begins with the highest to the lowest level qualifier.

Web Services scenario

- The following topics were updated or added to introduce the RESTful API— Web Services, How Client Programs Access the API, Software Requirements for Web Services, How to Configure and Deploy Web Services, Create or Update a Tomcat Directory Structure for the Web Services Component, Verify Your Apache Tomcat Installation and Configuration, How to Update an Existing Deployment of Web Services, How to Run Web Services Under New Releases of Tomcat, How Access to a Data Source Works, How to Create a Configuration File, V17 Data Source Configuration File Template,Using Web Services, CA Endevor SCM SOAP Web Services, How to Create a Client Stub,CA Endevor SCM RESTful API, RESTful API Client Program Design Considerations, RESTful API Outputs and Return Codes, RESTful API Action Request URIs,Repository Actions, List All Configurations, List Parameters of a Configuration, Package Actions, List Packages, Updated Packages, Cast Packages, Approve Packages, Deny Packages, Execute Packages, Backout Packages, Backin Packages, Commit Packages, Ship Packages, Delete Packages, Reset Packages, Inventory Actions, List Systems, List Subsystems, and List Elements.

Version 16.0, Fourth Edition

Package Ship scenario:

- Update Processors for Post-Ship Script (see page 130) and Create a Destination Configuration File (see page 132)— Updated to add that the destination specific symbols must have been created in the script data set members by your processors for substitution to take place correctly.

Version 16.0, Third Edition

Best Practice Implementation scenario:

- Configuration Tables (see page 275) and Run BPISTART (see page 277) — Added a note indicating that the WARN parameter should be set to YES in the configuration table BC1TNEQU.

Version 16.0, Second Edition

Web Services scenario:

- Create a Tomcat Directory Structure for the Web Services Component (see page 210)— Updated to change a reference to the target Tomcat directory from TARGET_TOMCAT_DIR=/sysname/cai/CASoftware/ESCM/tpv/tomcat to TARGET_TOMCAT_DIR=/sysname/cai/CADeploy/ESCM/tpv/tomcat.

Version 16.0

- This Scenario Guide was introduced with the Version 16.0 documentation set. This guide contains the following new or updated scenarios. These same scenarios are also listed as separate documents in the Knowledge Base Articles section of the bookshelf.
  - The following scenarios are new:
    - How to Enable Global Type Sequencing
    - How to Enable Web Services
    - How to Automatically Generate "Using" Elements with Autogen
    - How to Perform a Best Practices Implementation
  - The following scenarios are updated:
    - How to Enable and Secure Concurrent Action Processing
    - How to Set Up Package Ship

Contents

Chapter 5: Package Ship Facility 69

Chapter 1: Introduction

This guide contains role-based how-to scenarios for CA Endevor SCM.

Chapter 2: Global Type Sequencing

The following information describes how the CA Endevor SCM administrator creates a Type Sequence member and enables Global Type Sequencing.

This section contains the following topics:

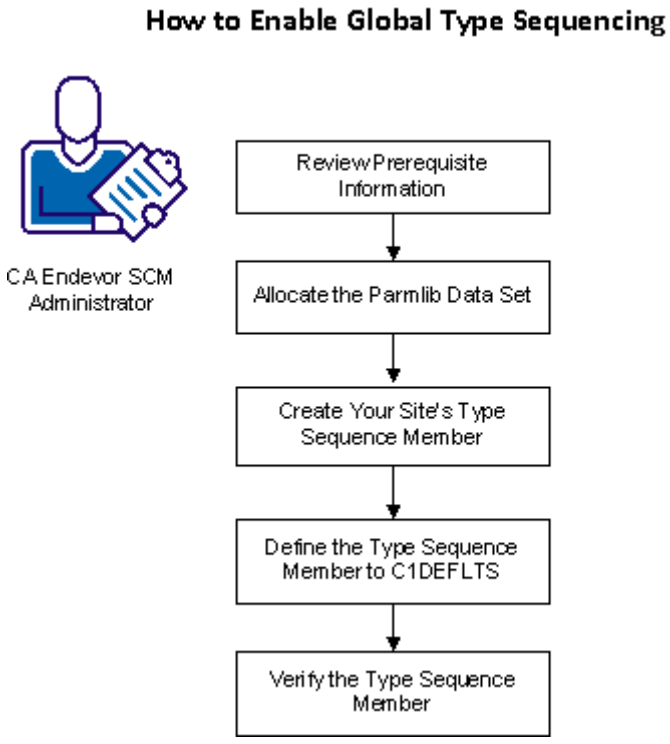
How to Enable Global Type Sequencing

As a change manager (CA Endevor SCM administrator), you can enable Global Type Sequencing to improve batch processing performance. When this feature is enabled, API element actions (if wildcarded or masked) and SCL element actions are processed by a single Type sequence defined at the site level in the Type Sequence member created by the administrator.

Global Type Sequencing is a prerequisite for the following features:

- Concurrent Action Processing (CAP)— When CAP is specified for a batch job or packages submitted for processing, certain element action requests are executed concurrently. CAP uses Global Type Sequencing to determine which actions can be processed at the same time.
- Autogen option for Add, Update, and Generate actions— When Autogen is specified for an element, elements that use this component element are automatically generated. Autogen requires Global Type Sequencing so that the elements that use a component are generated after all of the components have been processed (for example, macros before source, source before load modules).

The following graphic shows how to enable Global Type Sequencing.



To enable Global Type Sequencing, do the following:

1. Review Prerequisite Information (see page 15).
2. Allocate the Parmlib Data Set (see page 17)
3. Create Your Site Type Sequence Member (see page 18)
4. Define the Type Sequence Member to C1DEFLTS (see page 21)
5. Verify the Type Sequence Member (see page 22)

Review Prerequisite Information

Before enabling Global Type Sequencing, make sure that you understand how processing works. Review the following topics. These topics explain the difference between default Type processing and Global Type Sequencing and how Global Type Sequencing affects Concurrent Action Processing.

- Type Processing of Batch Jobs (see page 15)
- How Default Processing of Batch Jobs Works (see page 16)
- How Global Type Sequencing Works (see page 16)
- How Global Type Sequencing Affects Concurrent Action Processing (see page 16)

Type Processing of Batch Jobs

The CA Endevor SCM administrator can establish a processing sequence for CA Endevor SCM Types that must be processed in a specific order. For example, COBOL copybooks must be processed before COBOL programs. This Type processing sequence can be set either at the system level (the default) or at a global level, which is referred to as Global Type Sequencing. Other Types, such as JCL, can be processed in any order. You do not need to define such Types in the Type processing sequence.

When a batch request includes multiple element Types, CA Endevor SCM first builds two request chains. One chain is composed of sequenced Types in Type sequence order, and one chain is composed of nonsequenced Types in alphabetical order. CA Endevor SCM then processes the sequenced Type chain first, as follows.

- When default processing (without Global Type Sequencing) is used, Types are processed by Type order defined for each target System.
- When Global Type Sequencing is used, Types are processed according to the processing order defined at the site level.

Global Type Processing is a prerequisite for the Concurrent Action Processing option. This option enables certain Types to process at the same time.

How Default Processing of Batch Jobs Works

For default processing of batch jobs, Types are processed by Type order defined at the System level. Element actions execute within each System in the same order as the Types were defined to the System and Stage by the administrator or as reordered by the administrator. Details of this process follow:

1. As the administrator defines Types, they are added to the end of the Type sequence in the System MCF.
2. The administrator can reorder this sequence in foreground or batch as follows:
    a. The Type Sequence panel— You can use this panel to reorder the sequence only when Global Type Sequencing is not in effect.
    b. Define Type Sequence statement—You can use this Batch Administration SCL statement to reorder the sequence, whether or not Global Type Sequencing is in effect. However, if Global Type Sequencing is in effect, a warning message is generated, even though the action is still executed.
3. When batch jobs are submitted, element actions execute within each System in the sequence maintained in the target System's MCF.

How Global Type Sequencing Works

Global Type Sequencing is an optional method for batch processing. When Global Type Sequencing is used, API element actions (if wildcarded or masked) and SCL element actions are processed by a single Type sequence defined at the site level in the Type Sequence member created by the administrator. Details of this process follow:

1. The administrator enables Global Type Sequencing. This task includes defining the Type sequence order at the site level in a Type Sequence member.
2. When using Global Type Sequencing, element actions are processed by the Type sequence defined in the Type Sequence member.
3. 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. Those Types that are not included in the Type Sequence member are processed in the order in which they are defined in the target System's MCF record.

How Global Type Sequencing Affects Concurrent Action Processing

Global Type sequencing is required when using Concurrent Action Processing (CAP). You can make CAP more efficient by including a category definition for Types that can be processed simultaneously. Categories are defined in the Global Type Sequence table.

Important: Specifying a category in the Global Type Sequencing table does not affect how sequenced Type and nonsequenced Type requests are built and sorted. The category only affects how the actions are dispatched by Concurrent Action Processing.

CAP processes requests as follows:

1. When CA Endevor SCM is dispatching element actions to process concurrently, it refers both to the Type and to the category associated with the Type when deciding which action requests to dispatch.

   a. If the next request in the sequenced Type chain contains the same category value as the previous request, that element action can be dispatched to an available STC.

      With categories, CA Endevor SCM continues dispatching element actions from the sequenced Type chain as long as the category value matches the previous request. Available STCs only have to wait if sequenced Type requests from one category value are still processing and there are no non-sequenced Type requests to process.

   b. If the next request does not have the same category value, CA Endevor SCM attempts to dispatch a request from the nonsequenced Type chain.

      Without categories, element actions of the next sequenced Type number have to wait until all actions of the previous sequenced Type have completed processing. Nonsequenced Type element actions are then dispatched as STCs become available or after all the sequenced Type requests are dispatched. Available STCs remain idle if there are sequenced Type requests still processing, and there are no requests from the nonsequenced Type chain to process.

2. When all the requests have been processed for a particular sequenced Type and category, CA Endevor SCM begins processing requests for the next sequenced Type.

3. When all requests from the sequenced Type chain have been completed, any remaining requests from the nonsequenced Type chain are then dispatched.

Allocate the Parmlib Data Set

To contain the Type Sequence member, the Parmlib data set is required. To allocate the Parmlib data set, edit the BC1JEPRM member in your site's CA Endevor SCM JCLLIB data set, and then execute the batch job.

Note: If the Alternate ID feature is active, CA Endevor SCM uses the Alternate ID when accessing the PARMLIB file.

Create Your Site Type Sequence Member

To specify the Type processing sequence for Global Type Sequencing, your site needs a Type Sequence member. The order of the Type records in the Type Sequence member represents the processing sequence. Types not defined in the member are processed in the Type name order maintained in the target System MCFs after all Types defined in the Type Sequence file are processed. Complete the following steps:

Create the file using one of the following methods:

- Use the BC1JBTSE utility to build a Type sequence file to a sequential work file, and then edit this file.
- Use the standard ISPF editor. Add Type sequence records, as appropriate for your site.

Build a Type Sequence Member Using the BC1JBTSE Utility

To create a Type Sequence member for Global Type Sequencing, you can use the build utility BC1JBTSE.

Follow these steps:

1. Edit member BC1JBTSE to conform to the needs of your installation. Then execute this job. The build utility is member BC1JBTSE in your site's CA Endevor SCM CSIQJCL data set

   The JCL builds a Type sequence file to a sequential work file. The utility creates Type sequence records for all Types defined to a site. The utility reads the C1DEFLTS table to find the Master Control records and then extracts the necessary Type information from the MCF. For each unique Type name that is found, a Type sequence record is created.

   The utility does not merge one System's Type sequence with another. Unique Type records are written to the file as they are processed, starting with the first System in the first defined C1DEFLTS's environment, stage combination.

   The Type sequence record's description field contains the description from the first unique Type record. The utility produces comment records to identify each Type location occurrence. For example, if Type MAC is defined in environment DEV (stage 1, 2) under System BASE and in environment QA (stage 1,2) also under BASE, four comment records appear following the Type sequence record. The description field in the sequence record is set to the Type's description value.

   Note: If your site has environments that you want the utility to exclude, you can create a temporary C1DEFLTS table that excludes these environments and run the utility against the temporary C1DEFLTS. For example, you might have an environment with many test Systems that contain many elements that have never been promoted. Excluding this environment would limit the number of generated Type statements, which could help you define the initial Type sequence order. Any valid Types from the excluded environment could then be added manually.

2. Edit this file by rearranging the order of the Type record or by deleting Type records. The order of the records in the file will determine the order in which Types are processed.
3. Copy the file to the Parmlib data set.

Build a Type Sequence Record Using an Editor

The Type records in the Type Sequence member specify the Type processing order for Global Type Sequencing. You can create a Type sequence file using an ISPF editor.

Follow these steps:

1. Code Type records in an ISPF editor using the Type Record syntax (see page 19). Add Type records, as appropriate for your site.
2. Order the records in the order that you want the Types to process.

Type Record Syntax for Type Sequence File

The Type Record syntax defines the Type processing sequence in the Type Sequence member for Global Type Processing.

Type records use the following syntax:

```
TYPe 'Type-name' [SYStem 'system-name'] [CATegory 'category-value']
DESCription 'Sample Type Description'.
```

The following general rules apply to the Type record syntax:

- The ending period (.) delimiter is required.
- The Type name and description fields can be on separate lines.
- Each record is 80 bytes.
- No text can appear in columns 73 to 80.
- Comments can appear in the file by placing an asterisk in the first column position.

TYPE *Type-name*

Specifies a one- to eight-character Type. Enclosing the name in single quotes is optional. The name is converted to uppercase characters.

SYSTEM *system-name*

(Optional) Specifies the one- to eight-character System name where the Type is located. Enclosing the name in single quotes is optional. If Types with the same name are located in different Systems, this clause lets you specify different processing orders for each Type and System combination. If this clause is omitted, the Type sequence order for the Type name applies to the entire site. The name is converted to uppercase characters.

CATEGORY *category-value*

(Optional) Specifies a one- to eight-character alphanumeric name for a Type group. Enclosing the category-value in single quotes is required. During Concurrent Action Processing (CAP) dispatch, if Category was specified, the Category value is compared to the previous request. If the Category values match, the action is dispatched and processed simultaneously with the previous request.

The Category value does not affect how the request chains are built or sorted, nor does it affect normal processing order. This parameter simply enables the next Type in the sequence to start processing before the prior Type in the sequence finishes, provided both requests are in the same Category group. If the Category clause is omitted, the Type is still processed in order.

The value is converted to uppercase characters. If CAP is not enabled, the Category clause has no effect.

Note: The Type grouping capability affects processing only when CAP is in effect; it has no effect on Global Type Sequencing itself. Global Type Sequencing alone simply uses the Type processing order defined in the Type Sequencing member, but does not cause the simultaneous processing of actions.

DESCription *description*

Specifies an up to 50-character description of the Type. Enclosing the description in single quotes is required. When the text contains single quotes, these quotes are converted to blanks. The description text can be entered in either uppercase or lowercase characters.

Example: Global Type Sequence with Type Categories

In this example, the following table lists element Types in the order that a Administrator wants the Types to be processed. The Category column indicates the category for simultaneous processing for Concurrent Action Processing. Thus elements of Type ASMMAC, CHDR, and COPYBOOK can be processed at the same time when the Concurrent Action Processing feature is in use. Regardless of whether Concurrent Action Processing is being used, elements of Type CICSMAP cannot start processing untill all elements of Type COPYBOOK are finished.

Type	System	Category	Description
INCLUDE			Processor Includes
PROCESS			CA Endevor Processors
OPT			CONPARMX and Compiler Options
PARM			Utility Parameters
ASMMAC		1	Assembler Program Macros
CHDR		1	Program Headers
COPYBOOK		1	COBOL Copybooks

CICSMAP			CICS Maps
ASMPGM		2	Assembler Program Source
CPGM		2	C Program Source
COBOL		2	COBOL Program Source
FORTRAN		2	FORTRAN Source
JAVA		2	JAVA Source
PL1		2	PL/1 Source
EZTINCL			EZTRIEVE Includes
EZTRIEVE			EZTRIEVE Source
LINK			Linkage-Editor JCL
BIND			Data Base Binds

Define the Type Sequence Member to C1DEFLTS

To enable Global Type Sequencing, both the Parmlib data set and the Type Sequence member must be defined to the Defaults table C1DEFLTS. The Parmlib data set contains the Type Sequence member. The C1DEFLTS table will not assemble correctly if the Type Sequence member is defined to C1DEFLTS, but the Parmlib is not defined. An empty Type Sequence member will force Type sequencing by Type name order.

To define the Parmlib data set and the Type Sequence file, add the PARMLIB= and TYPESEQMBR= parameters to C1DEFLTS and then assemble and link-edit it to your authorized library as member C1DEFLTS.

Example: Define the Type Sequence member in C1DEFLTS

In the following example, the Parmlib data set and the Type Sequence member are defined in the C1DEFLTS table. The Parmlib data set name is BST.CASCMMF.PARMLIB and the Type sequence member is named ETYPESEQ.

```
Col 1       Col 16                                          Col 72
            C1DEFLTS TYPE=MAIN,                             X
            .
            .
            .
            PARMLIB='BST.CASCMMF.PARMLIB',                  X
            TYPESEQMBR=ETYPESEQ,                            X
            .
            .
            .
```

Note: The Environment Site Information panel displays Parmlib information on the second screen of the panel Site Information from the C1DEFLTS.

Verify the Type Sequence Member

After creating the Type Sequence member and identifying it to CA Endevor SCM in the C1DEFLTS table, you can test the syntax of the member.

Follow these steps:

1. Launch an CA Endevor SCM CLIST.

   If the Type Sequence table has been defined correctly, CA Endevor SCM starts. If there is a coding error in the table, console messages appear that identify the error. For example:

   ```
   + TYPE TYPE1 SYSTEM SYS1 CATEGORY 12345678A    DESCRIP
   +BSTPPARS: E006 VALUE SPECIFIED IS TOO LONG: 12345678A
   +h:mm:ss  B1TS011E  TYPE SEQUENCE TABLE HAS ERRORS
   +h:mm:ss  B1TS012E  SCL PROCESSING IS NOW DISABLED.
   ```

2. Correct any errors, and then launch the CLIST again. When CA Endevor SCM appears, open the Type Sequence display in foreground.

   The processing sequence displayed matches the sequence defined in the Type Sequence member.

Chapter 3: Concurrent Action Processing

The following information explains how to enable, understand, activate, and manage the Concurrent Action Processing (CAP) feature to speed the processing of batch jobs and packages.

To enable this feature, the CA Endevor SCM administrator requires the assistance of the following roles:

- CA Common Services administrator (or systems programmer)—This role is required to enable the CA Common Services Common Communications Interface (CAICCI) Spawn facility for CAP.
- Security administrator—This role is required to configure your site security software to allow the spawn facility to function for CAP. This role is also required if you want to limit who can activate CAP.

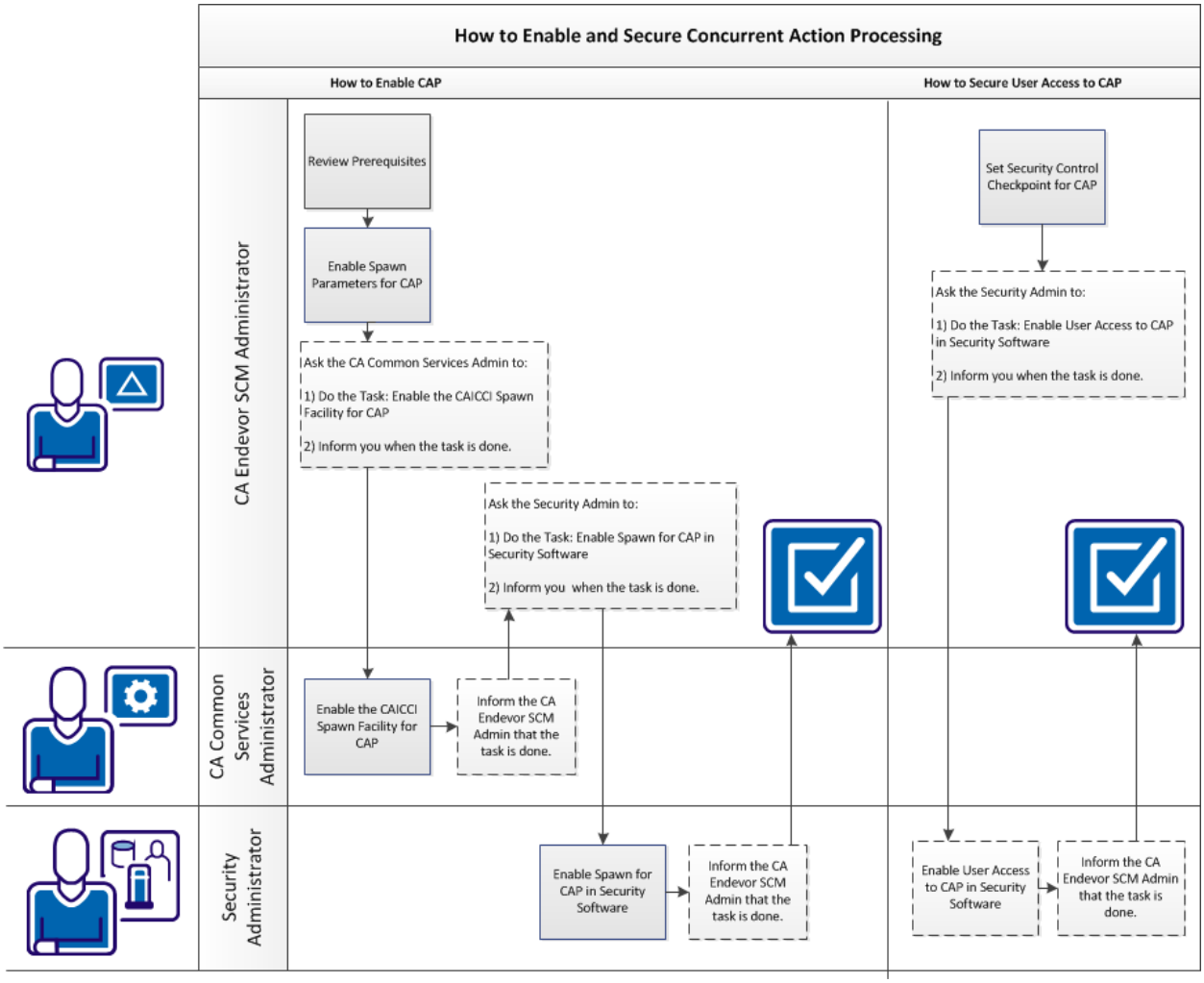
This section contains the following topics:

How to Enable and Secure Concurrent Action Processing

As a change manager (CA Endevor SCM administrator), you can enable the Concurrent Action Processing (CAP) feature to speed the processing of batch jobs and packages. CAP causes certain element action requests to be executed concurrently. This method of processing reduces the elapsed time that it takes to process multiple actions.

To complete the setup of this feature, you need the assistance of your CA Common Services administrator (or systems programmer) and your site security administrator. Optionally, you can limit who can use CAP.

The following diagram shows how you enable CAP and, optionally, how you secure user access to CAP.



How to Enable CAP

To enable CAP, the following roles and tasks are involved:

CA Endevor SCM administrator

This role completes the following steps:

1. Review Prerequisites (see page 27).
2. Enable Spawn Parameters for CAP (see page 32).
3. Ask your CA Common Services Administrator to perform the task Enable the CAICCI Spawn Facility for CAP. Also, ask them to let you know when they have successfully completed this task.
4. Ask your Security Administrator to perform the task Enable Spawn for CAP in Security Software. Also, ask them to let you know when they have successfully completed this task.

CA Common Services administrator

This role completes the following steps:

1. Enable the CAICCI Spawn Facility for CAP (see page 34)
2. Inform the CA Endevor SCM administrator that the CAICCI Spawn Facility for CAP task is done.

Security administrator

This role completes the following steps:

1. Enable the spawn facility for CAP in your security software. Depending on the security software in use at your site, complete one of the following tasks:
   - How to Configure CA Top Secret for CAP (see page 38).
   - How to Configure CA ACF2 for CAP (see page 39).
   - How to Configure IBM RACF for CAP (see page 40).
2. Inform the CA Endevor SCM administrator that your site security software is configured to enable the spawn facility CAP.

How to Secure User Access to CAP

Optionally, you can limit who can use CAP. To secure user access to CAP, the following roles and tasks are involved:

CA Endevor SCM administrator

This role completes the following steps:

1. Set a Security Control Checkpoint for CAP (see page 41)
2. Ask your security administrator to enable user access to CAP in your security software. Also, ask them to let you know when they have successfully completed this task.

Security administrator

This role completes the following steps:

1. Enable users read access to CAP in your security software—The security administrator enables user access in the site security software to the pseudo data set that the CA Endevor SCM administrator defined as the security checkpoint for CAP. Details on how to enable access to a data set are not included in the CA Endevor SCM documentation, because this is a usual procedure.
2. Inform the CA Endevor SCM administrator that the task to enable user access to the CAP data set is done.

Review Prerequisites

Before users can submit batch jobs or packages for Concurrent Action Processing, certain prerequisites are required. Complete the following prerequisites.

- Verify that Global Type Sequencing is enabled.

  Global Type Sequencing causes the batch processing of element actions to occur in a Type sequence defined at the site level. CAP uses this feature to determine which actions can be processed simultaneously. Global Type sequencing is required when using Concurrent Action Processing (CAP). You can make CAP more efficient by including a category definition for Types that can be processed simultaneously. For more information about Global Type Sequencing, see How Global Type Sequencing Affects Concurrent Action Processing (see page 16). For more information on enabling Global Type Sequencing, see the scenario How to Enable Global Type Sequencing.

- Before enabling CAP, the CA Endevor SCM administrator needs to understand how CAP works to properly set the spawn parameters. Review the following topics, which explain CAP processing:

  - Concurrent Action Processing (see page 28)
  - How Concurrent Action Processing Works (see page 29)

- Verify that the following CA Common Services components are installed and running on your z/OS mainframe that hosts CA Endevor SCM:

  - CAIENF—The Event Notification Facility enables concurrent action processing through the component CAICCI Communication Interface. If your site is running CAIENF r12 or higher, a database is not required for Concurrent Action Processing.
  - CAICCI—The Common Communications Interface is required only if you want to use Concurrent Action Processing, Web Services, the Eclipsed-Based UI, or CA CMEW.

  Note: CA Endevor SCM uses the CA Common Services Resource Initialization Manager (CAIRIM) component for product license authorization. CAIRIM is mandatory to run CA Endevor SCM, whether or not your site uses CAP or CA CMEW.

  Note: For more information about how to determine if the components are installed, see Verify CA Common Services are Running (see page 37).

- (Optional.) Enable IBM z/OS VSAM Record Level Sharing (RLS) facility to manage your CA Endevor SCM VSAM data sets. These data sets include all Master Control Files (MCFs), Element catalog and EINDEX, and the package data set.

  RLS improves VSAM performance and, so, improves the performance of CAP.

- Ensure that the C1DEFLTS referenced in the routing region JCL is the same as the C1DEFLTS referenced in the procedure JCL for the action request regions.

- Verify that your exits work as expected.

User exits are called in the routing region and action request regions. A summary of these exits is shown in the following table. For more information about exits, see the *Exits Guide*.

Exit Number	Where Called	When Called
1	Routing Region, Action Request Region	After security calls
2	Action Request Region Only	Before action is executed
3	Action Request Region Only	After action is executed
4	Action Request Region Only	Called before exit 2 for add, update, retrieve
5	Routing Region, Action Request Region	Called once in each job
6	Routing Region, Action Request Region	Called once in each CA Endevor SCM termination
7	Routing Region, Action Request Region	Called once in each CA Endevor SCM startup

Note: Certain requirements apply to the SCL requests in the jobs that are submitted for processing. Element names cannot be wildcarded. In addition, the To and From statements on all Add, Update, and Retrieve actions must reference cataloged data sets. For more information about SCL requests, see SCL Requirements for CAP (see page 48).

Concurrent Action Processing Overview

Concurrent Action Processing (CAP) speeds up the processing of batch jobs and packages. CAP causes certain element action requests to be executed concurrently, thus reducing the elapsed time that it takes to process multiple actions.

CAP uses Global Type Sequencing to determine which actions can be processed at the same time. The CAP facility processes certain element actions of the same type concurrently, and must complete the processing of all actions of the same type before starting to process elements of the next type. However, any action types that are not defined in the Global Type Sequence can be processed concurrently with any other action. The following actions can be processed concurrently: ADD, ALTER, UPDATE, RETRIEVE, MOVE, GENERATE, DELETE, SIGNIN, and TRANSFER CA Endevor SCM to CA Endevor SCM. All other actions on the request are processed serially.

The administrator can specify whether CAP is allowed and how many additional started tasks a single job can create for concurrent processing. These parameters are set in the C1DEFLTS table. Through the External Security Interface, the administrator can limit who is authorized to use CAP. This method of processing is not initiated automatically; users must request CAP at the job level.

Note: Concurrent action processing with CA CM Enterprise Workbench is available only for packages and enterprise packages, not for element actions.

How Concurrent Action Processing Works

Concurrent Action Processing (CAP) dynamically creates additional server address spaces to process certain actions concurrently. This processing method reduces the elapsed time that is required to process large numbers of actions. Thus CAP speeds the processing of batch jobs and packages.

When Concurrent Action Processing is requested, the request is processed in the following manner:

1. Any actions that are wildcarded or masked, are expanded and the actions are sorted in type sequence order to create a chain of element action requests.

   Note: Only use wildcarding or masking when you are submitting one action (for example, all Add actions) in the same job, to avoid unpredictable results.

2. Any actions that use an archive file as input, such as RESTORE, LIST from archive file, and TRANSFER, are executed serially in the routing region. These actions always run serially in the routing region.

3. Before execution of other actions, CA Endevor SCM determines whether CAP is enabled and the user is authorized in the following order:

   a. CA Endevor SCM determines whether Global Type Sequencing is enabled.

   b. If Global Type Sequencing is enabled, then the request JCL is examined for the presence of an EN$CAP or EN$CAPnn DD card. These parameters indicate whether the JCL requests Concurrent Action Processing.

   c. If the JCL requests CAP, then the C1DEFLTS parameters SPAWN and SPAWNCNT values are checked to see if CAP is enabled.

   d. A check is made through the External Security Interface to determine whether the user is authorized to request Concurrent Action Processing.

   If any of these conditions are *not* met, then all actions are processed serially.

4. If CAP was requested and is permitted as verified in step 3, the chain of action requests that remains from step 1 and 2 is examined. Any ADD, UPDATE, or RETRIEVE requests found in the chain are then preprocessed as follows.

   a. Any DDNAME is translated to a data set name; that is, the DDNAME is localized. These data set names are created in the following manner.

      - When a CA Endevor SCM server allocates a pseudo-temporary data set name, CA Endevor SCM builds the name using a jobnumber node in the CA Endevor SCM generated name, rather than the jobname. This is necessary to avoid data set contention among the action request regions.

- In some cases, CA Endevor SCM requests that the system generated names of temporary data sets be unique in the system. This is done to avoid data set contention among servers. Currently, when CA Endevor SCM allocates a temporary data set that is to have a system generated name, it does so by specifying a DALDSNAM text unit. This causes the temporary data set name to have the following form, which may **not** be unique:

  ```
  SYSyyddd.Thhmmss.RA000.jobname.dsname.Hnn
  ```

  However, in an action request region created for Concurrent Action Processing, CA Endevor SCM may eliminate the DALDSNAM for some temporary datasets. This causes a name to be generated with the following form, which is unique:

  ```
  SYSyyddd.Thhmmss.RA000.jjobname.Rggnnnnn
  ```

  Note: For temporary data sets created and deleted in the same processor step (that is, allocated with DISP=(NEW,DELETE,DELETE), or with no DISP at all), the DALDSNAM is not eliminated. This means that such data sets are **always** generated as SYSyyddd.Thhmmss.RA000.jobname.dsname.Hnn, even with Concurrent Action Processing. Therefore, contention is still possible with such data sets.

b. The data set specified (using the DSNAME or DDNAME parameter) is checked to determine if it is a cataloged data set. If any data set refers to an uncataloged file (for example, a SYSIN/SYSOUT data set, a temporary data set or a non-SMS data set with a disposition of NEW), then Concurrent Action Processing cannot be enabled and the actions will be processed serially.

Important: If you use exit 4 programs to alter input and output files before the execution of an ADD, UPDATE, or RETRIEVE action, the data sets specified in the SCL must be cataloged data sets. This requirement applies even if the user has an exit 4 program that changes the data set names.

5. Action request regions are created equal to the SPAWNCNT value set in the C1DEFLTS table. However, if the SPAWNCNT value was overridden on the job request, that value is used to determine the maximum number of action request regions. The value can be changed for a specific job in the following manner:

   - For batch requests, by using the EN$CAPnn DD card on a batch job request JCL.
   - For foreground requests, by using the Concurrent Number field on the Submit Package panel or the Batch Options Menu.

   Note: If CA Endevor SCM is called from a processor, it will not create any servers. In addition, if an action request region is created during concurrent action processing, it will not create additional servers.

6. The actions ADD, ALTER, UPDATE, RETRIEVE, MOVE, GENERATE, DELETE, SIGNIN, and TRANSFER (CA Endevor SCM to CA Endevor SCM) are dispatched to action request regions. Actions that can be processed simultaneously are routed from the routing region of the originating job to the action request regions and processed concurrently.

   - The *routing region* is the batch job, TSO session, or CA CM Enterprise Workbench session that has requested that actions be processed concurrently.
   - *Action request regions* are the started tasks that are created to process the actions. Multiple action request regions are created. They receive individual action requests from the routing region, process them, and send the results back.

   CAP uses Global Type Sequencing to determine which actions can be processed at the same time. Global Type Sequencing determines whether actions can be processed concurrently. Elements in the same type sequence can be processed concurrently. Additionally, elements that are not listed in the global type sequence can be processed concurrently with any element. There is no guarantee that actions on elements in the same type sequence (or on elements that are not type sequenced) will be processed in the order in which they appear in the SCL. However, actions against the same element name will be processed serially and in the order in which they appear in the SCL, provided the element names are explicit. One action must complete before the next is dispatched.

   Note: If you specify AUTOGEN, SCL requests are fully resolved based on current inventory contents before any actions are processed. For example, suppose that you have a GENERATE action with the AUTOGEN option and several MOVE actions in the same request. The MOVE actions will not include any of the additional elements that are copied back as a result of the GENERATE action.

7. After all other actions have completed processing, the actions PRINT, ARCHIVE, LIST, and TRANSFER to archive data set are performed serially in the routing region.

How Global Type Sequencing Affects Concurrent Action Processing

Global Type sequencing is required when using Concurrent Action Processing (CAP). You can make CAP more efficient by including a category definition for Types that can be processed simultaneously. Categories are defined in the Global Type Sequence table.

Important: Specifying a category in the Global Type Sequencing table does not affect how sequenced Type and nonsequenced Type requests are built and sorted. The category only affects how the actions are dispatched by Concurrent Action Processing.

CAP processes requests as follows:

1. When CA Endevor SCM is dispatching element actions to process concurrently, it refers both to the Type and to the category associated with the Type when deciding which action requests to dispatch.

   a. If the next request in the sequenced Type chain contains the same category value as the previous request, that element action can be dispatched to an available STC.

      With categories, CA Endevor SCM continues dispatching element actions from the sequenced Type chain as long as the category value matches the previous request. Available STCs only have to wait if sequenced Type requests from one category value are still processing and there are no non-sequenced Type requests to process.

   b. If the next request does not have the same category value, CA Endevor SCM attempts to dispatch a request from the nonsequenced Type chain.

      Without categories, element actions of the next sequenced Type number have to wait until all actions of the previous sequenced Type have completed processing. Nonsequenced Type element actions are then dispatched as STCs become available or after all the sequenced Type requests are dispatched. Available STCs remain idle if there are sequenced Type requests still processing, and there are no requests from the nonsequenced Type chain to process.

2. When all the requests have been processed for a particular sequenced Type and category, CA Endevor SCM begins processing requests for the next sequenced Type.

3. When all requests from the sequenced Type chain have been completed, any remaining requests from the nonsequenced Type chain are then dispatched.

Enable Spawn Parameters for CAP

The CAP option requires that you enable spawn parameters in the Defaults table (C1DEFLTS). CAP uses the parameters to spawn address spaces that allow multiple element actions to be processed at the same time.

Follow these steps:

1. Open the C1DEFLTS member located in the *iprfx.iqual*.CSIQSRC.
2. Change SPAWN=N to SPAWN=Y in the TYPE=MAIN section of the C1DEFLTS table.

   The spawn function for CAP is set to enable CAP.

3. Set the following parameters in the C1DEFLTS table.

   SPAWNCNT=*n*

   Specifies the default number of spawned regions that are started when CAP is used. Valid values are 2 through 99, but it cannot exceed the value of the parameter SPAWNMAX. The field can also be set to 0 to disable CAP when SPAWN=Y. If SPAWN=N, SPAWNCNT must be set to 0.

   SPAWNMAX=*n*

   Specifies the maximum number of tasks that a single job can spawn. Valid values are 2 through 99. The field can also be set to 0 to disable CAP if SPAWN=Y. If SPAWN=N, SPAWNMAX must be set to 0.

   Note: The CA Common Communications Interface (CAICCI) SERVICE definition MAX#_PROCESSES can be used to limit the total number of service processes on a system.

   SPAWNPROC=*CAP_server_name*

   Specifies the one to eight-character name of the CAP server. This name must be a CAICCI SERVICE definition service_name and a started task procedure name. A name is only required if the SPAWN=Y.

   Note: If SPAWN=Y and SPAWNCNT=0, then CAP is disabled. However, users can override the SPAWNCNT value for a specific job by using the EN$CAP*nn* DDNAME on the job request JCL. However, users cannot override the SPAWNMAX value.

   Then press End.

   Your changes to C1DEFLTS are saved. The spawn parameters are set according to your preferences. The Edit session ends and the Edit Entry Panel opens.

4. Complete the fields to select the BC1JTABL member located in the installation library *iprfx.iqual*.CSIQJCL. Press Enter.

   The sample JCL BC1JTABL opens in an ISPF Edit panel.

5. Edit the sample JCL BC1JTABL found in *iprfx.iqual*.CSIQJCL to your installation standards. Execute the job.

   The Defaults table is assembled, link-edited, and stored as member C1DEFLTS in your authorized user library, iprfx.iqual.CSIQAUTU. The spawn parameters are enabled for CAP processing. However, CAP cannot be used until the CAICCI Spawn Facility is enabled for CAP.

Note: Instead of steps 4 and 5, you can assemble and link-edit C1DEFLTS using an SMP/E USERMOD.

Example: C1DEFLT Spawn Parameters for CAP

In this example, the C1DEFLTS table's spawn parameters are set to enable CAP. The spawn facility for CAP is activated, because SPAWN=Y. When CAP is used, CAICCI uses the spawn procedure ENDEVOR. Eight spawn regions are started and one job can spawn a maximum of 99 address spaces.

```
SPAWN=Y                 ACTIVATE THE SPAWN FACILITY
SPAWNCNT=8,             SPAWN COUNT
SPAWNMAX=99,            MAXIMUM SPAWN COUNT
SPAWNPROC=ENDEVOR,      NAME OF SPAWN PROC
```

Note: Ask your CA Common Services administrator (or systems programmer) to enable the CAICCI Spawn Facility for CAP. Also ask the administrator to let you know when they have successfully completed this task.

Enable the CAICCI Spawn Facility for CAP

CAP uses the CA Common Services Common Communications Interface (CAICCI) Spawn facility. This facility spawns multiple address spaces to enable multiple actions to be processed at the same time. The following setup steps are required to use the spawn facility:

- A CAP spawn parameter definition file must be appended to the SPNPARMS DD in the CA Event Notification Facility (CAIENF) procedure.
- The spawn parameter file must be customized. The customization specifies the started task to enable the spawn facility to spawn address spaces in the z/OS environment where CA Endevor SCM is installed.

Follow these steps:

1. Open the CAIENF member located in your CAI.CAIPROC library.

   Note: If CAIENF is not in the CAI.CAIPROC library, your system administrator could have copied the CAIENF member to your system PROCLIB.

2. Append the CAP spawn parameter file definition named SPNPARMS DD to the CAIENF procedure JCL. For example, in the sample JCL below, you add and customize the two lines in bold.

```
//ENF PROC OPTLIB='SYS2.CA90S.PARMLIB',
//                  ENFDB='SYS2.CA31.ENFDB',
//                  ENFPARM=ENFPRM31,
//                  SPNPAR1=SPWNSNMP,
//                  SPNPARn=ndvspawn,
//                  CCIPARM=CCIPCA31,
//                  ENFCMDS=ENFCMD31,
//ENF      EXEC PGM=CAS9MNGR,TIME=1440
//CASRT01  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//CASRT02  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//CASRT03  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//CASRT04  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//CASRT05  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//SPNPARMS DD DISP=SHR,DSN=&OPTLIB(&SPNPAR1)
//         DD DISP=SHR,DSN=&OPTLIB(&SPNPARn)
//SPNPRINT DD SYSOUT=X
//SPNDEBUG DD SYSOUT=X
//SRVDEBUG DD SYSOUT=X
//ENFDB DD DISP=SHR,DSN=&ENFDB
//ENFPARMS DD DISP=SHR,DSN=&OPTLIB(&ENFPARM)
//         DD DISP=SHR,DSN=&OPTLIB(&CCIPARM)
//ENFCMDS  DD DISP=SHR,DSN=&OPTLIB(&ENFCMDS)
//SYSPRINT DD SYSOUT=X
```

SPNPAR*n*

Specifies a symbolic for a CAIENF parameter member. The value for *n* must be the next sequential number in the list of parameter files for the spawn function.

ndvspawn

Specifies the spawn parameter file for CAP. Throughout this procedure, this file is named ndvspawn. When you save the member in step 4, you can name the member as appropriate for your site. *However, ensure that you substitute that name whereever ndvspawn is referenced in this procedure, including here in the SPNPARMS DD statement.*

Then press End.

Your changes are saved. The CAP spawn parameter file is defined to the CAIENF procedure for CAICCI. The Edit session closes, and the Edit Entry panel opens.

3. Open the CAPCCI member, located in the installation source library *iprfx.iqual*.CSIQOPTN, in an ISPF Edit panel.

4. Customize the spawn parameters by editing the following CAICCI SERVICE and PROCESS statements in the CAPCCI file. Verify that the procname and the task name are the same. Keep the statements in the exact format provided in the sample file, maintaining spaces and column alignments.

```
*********************************************************************
           * CCI SERVICE STATEMENTS FOR ENDEVOR

*********************************************************************
           ENDEVOR      SERVICE SERVER_NAME=MVS_START_SERVER,
                                DEALLOCATE=TERMINATE,
                                LOST_CLIENT=DEALLOCATE,
                                MAX#_SERVICES=100,
                                MAX#_CLIENTS=1,
                                MAX#_PROCESSES=100,
                                SERVICE_UNAVAILABLE=START_SERVICE,
                                START=SPAWN_ONLY,
                                SIGNON/NOPASSCHK=SERVICE

*********************************************************************
           * CCI PROCESS STATEMENTS FOR ENDEVOR

*********************************************************************
                               PROCESS PROCESS_TYPE=MVS_STC,
                               PROCNAME=ENDEVOR,
```

SERVICE statement

Specifies the host application.

PROCESS statement

Specifies the JCL procedure that executes CA Endevor SCM.

PROCNAME=*procname*

Specifies the name of the started task procedure that the Spawn facility initiates. For example, PROCNAME=ENDEVOR.

Then enter the Save command and specify a name for the spawn parameter file for CAP (ndvspawn).

The CAP spawn parameter file (ndvspawn) is customized to specify the started task procedure that CAICCI can use to spawn address spaces for CAP. Your changes are saved. The Edit session closes, and the Edit Entry panel opens.

5. Save the customized file in your CAI.PARMLIB data set as the member *ndvspawn.*

   The file is now saved in a location where CAICCI can use it to spawn address spaces for CAP.

6. Open the ENDEVOR member, located in the CSIQJCL library that is delivered with CA Endevor SCM, in an ISPF Edit panel.

7. Edit the sample ENDEVOR started task procedure as appropriate for your site. Then enter Copy on the command line and specify the PROCLIB defined to JES where you want to copy the file.

   Your changes are saved. The started task procedure ENDEVOR is available to spawn the address spaces where multiple element actions can be processed at the same time. The Edit session closes, and the Edit Entry panel opens.

8. Recycle CAICCI or perform an IPL.

   The changes to CAICCI are in effect. Therefore, the CAICCI SPAWN facility is in effect for CAP and CA Endevor SCM can process jobs submitted with the CAP option. When users submit a batch job or package from foreground panels, the option to specify CAP is available on the panel.

Note: If you decide to use a PROC name other than ENDEVOR, update the other locations where the PROC name is used. Verify the SPAWNPROC parameter in the C1DEFLTS table and the CAICCI Service name and CAICCI PROC in the PROCLIB member all contain the same value. This parameter is also the same name as the PROCNAME= value specified in the CAICCI definitions.

You have now successfully enabled the Concurrent Action Processing (CAP) feature to speed the processing of batch jobs and packages.

Note: Inform your CA Endevor SCM administrator to let them know that the CAICCI SPAWN facility is in effect for CAP.

Verify CA Common Services are Running

The CA Common Services components CAIRIM, CAIENF, and CAICCI must be installed and running on your mainframe. CAIENF and CAICCI are required only if you want to use the Concurrent Action Processing feature, the Web Services component, or the companion product CA CMEW. If they are not already installed and running, you must install these services according to the instructions in the *CA Common Services for z/OS Getting Started.*

To determine whether CAIENF and CAICCI are installed and running

1. Go to the SDSF status display and enter the prefix ENF*.

   Note: If you do not see ENF* active on your system, contact your systems programmer to start it.

2. If the job is running, select it and search for (Find) CAICCI within the job.

   - If you find CAICCI, then CAIENF and CAICCI are running.
   - If you do not find CAICCI, then check your site's CAI.CAIPROC library to see if the ENF member has been modified to run CAICCI at your site.

     If the ENF member does not exist, then contact your CA Common Services administrator (or systems programmer) to determine the status of CAIRIM, CAIENF, and CAICCI on your system.

Setting Up Security for CAP

Your site's security administrator must enable the spawn facility for CAP in your security software, depending on the security software in use at your site. The CA Endevor SCM administrator must set the security control checkpoint for CAP in CA Endevor SCM. These tasks are discussed next.

How to Configure CA Top Secret for CAP

For the CA, Inc. Event Notification Facility (CAIENF) to spawn the SPAWNPROC defined to the C1DEFLTS table, the task must be defined to your security software package with a corresponding default user ID. Using these definitions, CAIENF will start the spawned task. The task will start under the security context of the default user ID, but then switch security context to that of the user submitting the CA Endevor SCM job.

Important: In the following description ENDEVOR is used as both the SPAWNPROC started task name and its corresponding user ID. If your site already has user ID ENDEVOR defined as the alternate user ID, do not use ENDEVOR for your task name or task user ID. Instead, select a different value. The alternate ID (ALTID) is defined in the C1DEFLTS table as RACFUID=ENDEVOR.

If your site uses CA Top Secret security, complete the following steps:

1. Define a new facility name ENDEVOR for CAP. To do this, add the following definitions to the Top Secret Security parameter file (specified by the PARMFIELD DD statement):

   ```
   * USERnn FACILITY FOR
   *
   FAC(USERnn=NAME=ENDEVOR)
   FAC(ENDEVOR=xxxx)
   ```

   Where 'xxxx' is any other facility control options to be set other than the defaults. The facility can be defined dynamically using the TSS MODIFY command. For example:

   ```
   TSS MODIFY(FAC(USERnn=NAME=ENDEVOR))
   TSS MODIFY(FAC(ENDEVOR=xxxx))
   ```

   The TSS MODIFY command is only valid until the next recycle of CA Top Secret.

2. Define a new ACID named ENDEVOR by entering the following command:

   ```
   TSS CRE(ENDEVOR) NAME(endevor user-id?) TYPE(USER) FAC(STC,ENDEVOR) PAS(xxxx,0)
   TSS ADD(ENDEVOR) FAC(STC)
   ```

   CA Top Secret recommends that all started task (STC) acids be given a password and that OPTIONS(4) be set in the CA Top Secret parameter file. OPTIONS(4) eliminates the prompt for a password when the STC starts. However, if someone tries to log on with the STC acid, that person will need to know the password.

   The NODSNCHK, NORESCHK, and NOSUBCHK bypass attributes on the ENDEVOR ACID may be required. If not, verify that the ACID is authorized to access all the files and resources it requires.

3. Give the ENDEVOR ACID a MASTFAC definition by entering the following command:

   ```
   TSS ADD(ENDEVOR) MASTFAC(ENDEV0R)
   ```

4. Assign userid ENDEVOR as the default ACID for the CAICCI spawned task ENDEVOR by entering the following command:

   ```
   TSS ADD(STC) PROCNAME(ENDEVOR) ACID(ENDEVOR)
   ```

5. Grant each user of CA SCM for Mainframe access to the ENDEVOR facility by entering the following command:

   ```
   TSS ADD(USER-ID) FAC(ENDEVOR
   ```

   Note: For more information about defining a new facility, see the *CA Top Secret Security Control Options* guide. For more information about the CRE and ADD commands, see the *CA Top-Secret Security Command Functions* guide.

How to Configure CA ACF2 for CAP

For the CA, Inc. Event Notification Facility (CAIENF) to spawn the SPAWNPROC defined to the C1DEFLTS table, the task must be defined to your security software package with a corresponding default user ID. Using these definitions, CAIENF will start the spawned task. The task will start under the security context of the default user ID, but then switch security context to that of the user submitting the CA Endevor SCM job.

Important: In the following description ENDEVOR is used as both the SPAWNPROC started task name and its corresponding user ID. If your site already has user ID ENDEVOR defined as the alternate user ID, do not use ENDEVOR for your task name or task user ID. Instead, select a different value. The alternate ID (ALTID) is defined in the C1DEFLTS table as RACFUID=ENDEVOR.

To configure CA ACF2 for CAP, complete the following steps:

1. Create an STC logon ID named ENDEVOR for the Concurrent Action Processing Started task by entering the following commands:

   ```
   ACF
   INSERT ENDEVOR NAME(ENDEVOR) STC
   ```

2. Verify that the ENDEVOR logon ID is defined with site-specific logon ID fields such as those fields used to create the UID string.
3. Verify that the ENDEVOR logon ID has access to all the data sets in the ENDEVOR task by writing CA ACF2 security ACCESS rules for the logon ID.

Note: For more information about configuring CA ACF2, see the *ACF2 Security Administration Guide* or contact CA ACF2 Technical Support.

How to Configure RACF for CAP

For the CA, Inc. Event Notification Facility (CAIENF) to spawn the SPAWNPROC defined to the C1DEFLTS table, the task must be defined to your security software package with a corresponding default user ID. Using these definitions, CAIENF will start the spawned task. The task will start under the security context of the default user ID, but then switch security context to that of the user submitting the CA Endevor SCM job.

Important: In the following description ENDEVOR is used as both the SPAWNPROC started task name and its corresponding user ID. If your site already has user ID ENDEVOR defined as the alternate user ID, do not use ENDEVOR for your task name or task user ID. Instead, select a different value. The alternate ID (ALTID) is defined in the C1DEFLTS table as RACFUID=ENDEVOR.

To customize IBM RACF to allow the Concurrent Action Processing started task to initialize correctly, complete the following steps:

1. Define a started task to RACF, using either of the following methods:
   - Define a new profile to the STARTED class (recommended by IBM)
   - Add a new entry in the started procedures table (ICHRIN03)

   Note: For more information, see the *IBM RACF Security Administrator Guide*.

2. Assign a RACF user ID to the started task xxxxxxxx and assign the user ID to a RACF group authorized to initiate started procedures. To define a RACF user ID for xxxxxxxx, use the ADDUSER command and associate it with your existing started task RACF group, as follows:

   ```
   ADDUSER user_name DFLTGRP(default_group) OWNER(default_group) NOPASSWORD
   ```

   user_name

   Specifies the name of the new RACF user ID. This name should be the same as the name of the started task member in your PROCLIB that CAP uses.

   default_group

   Specifies the default group that contains all system started tasks; for example, STCGROUP.

   Note: This command is only an example. For more information about using the ADDUSER command, see your RACF administrator.

   Note: If you do not know the name of the default group, see your RACF administrator. For detailed information to implement the RACF STARTED class or to modify the started task table (ICHRIN03), see the *IBM RACF Security Administrator Guide*.

Set a Security Control Checkpoint for CAP

You can set a security control checkpoint for CAP, by defining a pseudo data set in the CA Endevor SCM External Security Interface (ESI). A pseudo data set represents a data set access rule. The pseudo data set does not refer to a physical data set.

The security control checkpoint works with your site security software to determine whether users have authority to use CAP. Your security software must allow users read access to the pseudo data set to authorize users to use CAP. If a user does not have authority to use CAP, actions are processed in the order they are defined for Global Type Sequencing.

Follow these steps:

1. Open the External Security Interface (ESI) Name Equates Table, member BC1TNEQU, located in your *iprfx.iqual*.CSIQSRC installation library.

2. Add a NAMEQU CONCURRENT_ACT_PROC entry and specify the L1 and L2 values.

```
NAMEQU CONCURRENT_ACT_PROC.
     L1=('high_level_qualifier'),
     L2=('next_level_qualifier')
```

NAMEQU CONCURRENT_ACT_PROC

Specifies the pseudo data set name that the RACROUTE macro uses to secure CAP processing. Set the following values:

L1

Specifies the high-level qualifier for the data set.

L2

Specifies the next-level qualifier for the data set. We recommend that you specify a value of CAP to make it easy to understand the purpose of pseudo data set.

Then press End.

Your changes are saved. A pseudo data set name is defined in your ESI Name Equates Table to act as a security checkpoint for CAP. The Edit session closes, and the Edit Entry panel opens.

3. Assemble and link-edit member BC1TNEQU.

Now when users request access to CAP, the pseudo data set defined in the External Security Interface (ESI) acts as a security control checkpoint. If your site security software allows users read access to the pseudo data set, the users have authority to use CAP.

Note: Ask your security administrator to grant read access to the pseudo data set for the users you want to have access to CAP. Also ask the security administrator to let you know when they have successfully completed this task.

Example: A sample CONCURRENT_ACT_PROC rule for Name Equates Table

In this example, the Name Equates table includes the following entry.

```
NAMEQU CONCURRENT_ACT_PROC.
      L1=('C1'),
      L2=('CAP')
```

Users who have read authority to the data set library C1.CAP have authority to request CAP.

How to Activate Concurrent Action Processing

To speed up the processing of batch jobs and packages, you can use the Concurrent Action Processing (CAP) option. To activate CAP, an EN$CAP or EN$CAPnn DD statement is required in the job step that executes CA Endevor SCM. In addition, the SCL statements in the batch jobs or packages that are submitted for CAP processing, must conform to certain requirements.

To add the DD statement and activate CAP, use one of the following methods:

- Activate CAP in Batch JCL (see page 44)
- Activate CAP for Packages Submitted in Foreground (see page 46)
- Activate CAP for Batch Jobs Submitted in Foreground (see page 47)

For more information about SCL requirements, see SCL Requirements for CAP (see page 48).

Note: Before you can use CAP, this option must be enabled by your CA Endevor SCM administrator. For more information about enabling CAP, see How to Enable and Secure Concurrent Action Processing (see page 23).

Activate CAP in Batch JCL

To speed the processing of batch jobs and packages that contain a large number of actions, you can request Concurrent Action Processing in the JCL that executes CA Endevor SCM.

Follow these steps:

1. Verify that the SCL requests meet these requirements:
   - All elements are fully qualified; wildcarding is not used for elements.
   - On any Add, Update, or Request statements, the referenced data sets are cataloged data sets.

   Note: For more information about SCL requirements, see SCL Requirements for CAP (see page 48).
2. Add an EN$CAP or EN$CAP*nn* DD statement in the job step that executes CA Endevor SCM.

   Note: When you submit a job or package for batch processing from foreground panels, the option to specify concurrent action processing is provided on the panel. When you select this option, the EN$CAPnn DD statement is automatically included in the submitted JCL.

   ```
   //EN$CAP      DD SYSOUT=*
   ```

   Specifies that the number of Concurrent Action Processing processors that are allowed for this job is equal to the SPAWNCNT value in C1DEFLTS. For Concurrent Action Processing to occur, SPAWN=Y must be coded in C1DEFLTS and SPAWNCNT must be greater than 1.

```
//EN$CAPnn DD SYSOUT=*
```

nn

Specifies the maximum number of action request regions that are allowed for a job. This parameter limits the Extended Common Storage Area (ECSA) that is used by CAP.

00 – Concurrent action processing will not be used and actions will be processed sequentially.

02-nn – Indicates the maximum number of action request regions that are allowed per job. This value overrides the SPAWNCNT value in C1DEFLTS, but does **not** override the SPAWNMAX value in C1DEFLTS. SPAWN=Y must be coded in C1DEFLTS to enable Concurrent Action Processing. To determine what *nn* value to specify, use the following formulas. Where x in the formula is the same as the nn value for EN$CAP*nn*.

- To determine the minimum amount of ECSA that must be free before CAP is started,use the following formula:

  Minimum ECSA = 3.6K*x + 3.2K

- To determine the maximum amount of ECSA that a CAP job can use, use the following formula:

  Maximum ECSA = 40.1K*x + 3.2K

- To determine the size of the ECSA storage area that a CAP job can use that will not use all the ECSA free storage, add 20% to the maximum value, as follows:

  Maximum ECSA plus 20% = (40.1K*x + 3.2K)1.20

Your processing preferences are set. When you submit the job, it will be processed according to your preferences. If more than one EN$CAP*nn* card is included in the JCL for the job step, only the first one is used.

Activate CAP for Packages Submitted in Foreground

To speed the processing of batch jobs and packages that contain a large number of actions, you can request Concurrent Action Processing in the JCL that executes CA Endevor SCM.

Follow these steps:

1. Verify that the SCL requests meet these requirements:
   - All elements are fully qualified; wildcarding is not used for elements.
   - On any Add, Update, or Request statements, the referenced data sets are cataloged data sets.

   Note: For more information about SCL requirements, see SCL Requirements for CAP (see page 48).
2. Edit the following options on the *Submit Package* panel.

   Note: If CAP is not enabled, the following options are grayed out on the panel.

   Concurrent Action Processing

   Use this field to indicate whether you want to use concurrent action processing. Valid values are Y and N. The default value when you enter the panel is N. If this feature is not enabled for your site, this option is read-only.

   Concurrent Number

   Specify the number of concurrent actions that are to be processed, if you are using concurrent action processing. The default is the SPAWNCNT value set in C1DEFLTS. If you overwrite the default and then decide you want to use the default, either type in the default value or blank out this field. Valid values are 02 through the Max number that is shown on the panel. The Max number is the value of SPAWNMAX specified in the C1DEFLTS.

   Your processing preferences are set. When you submit the job, it will be processed according to your preferences.

Activate CAP for Batch Jobs Submitted in Foreground

When you submit a job for batch processing from the foreground Submit Package panel, you can request concurrent action processing.

Follow these steps:

1. Verify that the SCL requests meet these requirements:
   - All elements are fully qualified; wildcarding is not used for elements.
   - On any Add, Update, or Request statements, the referenced data sets are cataloged data sets.

   Note: For more information about SCL requirements, see SCL Requirements for CAP (see page 48).
2. Edit the following options on the *Batch Options Menu*:

   Concurrent Action Processing

   Use this field to indicate whether you want to use concurrent action processing. Valid values are Y and N. The default value when you enter the panel is N. If this feature is not enabled for your site, this option is read-only.

   Concurrent Number

   Specify the number of concurrent actions that are to be processed. The default is the SPAWNCNT value set in C1DEFLTS. If you overwrite the default and then decide you want to use the default, either type in the default value or blank out this field. Valid values are 02 through the Max number that is shown on the panel. The Max number is the value of SPAWNMAX specified in the C1DEFLTS.

   Your processing preferences are set. When you submit the job, it will be processed according to your preferences.

SCL Requirements for CAP

When batch jobs or packages are submitted for CAP processing, the SCL statements in these jobs must conform to the following requirements to process properly.

- Use explicit element names in SCL, if you plan to submit different actions in the same job.

  If you use wildcards in the element or member names, then actions against the same element may not be processed in statement order. For example, if you code the following, with the GEN following the ADD statement, the ADD may not complete before the GEN begins:

  ```
  ADD '*' TO ENV 'TESTENV' SYSTEM 'TESTSYS' SUBSYSTEM 'TESTSBS' TYPE 'TESTTYPE'…
  GEN '*' FROM ENV 'TESTENV' SYSTEM 'TESTSYS' SUBSYSTEM 'TESTSBS' TYPE 'TESTTYPE'
  …
  ```

- You can use wildcards or masking, if you submit one action per job. For example, you can use wildcards when submitting multiple Add actions in the same job.
- Examine your SCL requests. If your job will use Concurrent Action Processing, check all ADD, UPDATE, and RETRIEVE actions ensuring that all data sets referenced by the TO FILE, TO DSNAME, TO DDNAME and FROM FILE, FROM DSNAME, and FROM DDNAME are cataloged data sets. They cannot be uncatalogued, temporary or JES2 file data sets.
- If your site uses exit 4 programs to alter input/output files prior to the execution of an ADD, UPDATE, or RETRIEVE, ensure that the data sets specified in the SCL are cataloged data sets, even if the exit 4 program changes the data set names.

How to Monitor Concurrent Action Processing

To monitor Concurrent Action Processing (CAP), the CA Endevor SCM administrator can use MVS modify commands to view action status (see page 49).

View Action Status

You can issue MVS modify commands to the routing region in order to determine the status of the actions. Reponses are written to the region's joblog.

The general format of an MVS modify command is as follows:

F *jobname.identifier,parameters*

D[ISP] STAT

D[ISP] STATS

D[ISP] STATISTICS

Displays statistics.

A display statistics command returns the following response:

```
Number of Completed Requests         nnnnnn
Number of Executing Requests         nn
Number of Action Request regIons     nn
Max Elapsed Time so far              nnnnnn  Act# num Stmt #num action element
type
Min Elapsed Time so far                      nnnnnn  Act #num Stmt #num action
element type
Total Elapsed Time collected so far  nnnnnn
```

D[ISP] ACTIVE REQS

D[ISP] ACTIVE REQUESTS

Displays status of currently active action requests. This parameter returns the following response:

```
ACT# num Stmt #num ACTIVE ON jobname jobnumber cci-id action element type.
```

D[ISP] SERVERS

Display the status of spawned servers. If the server is busy processing a request, we will display information about the request. This parameter returns the following response:

```
REGION jobname jobnumber cci-id status [ACT# nnnn STMT# nnnn action element-type]
```

The cci-id status is BUSY, AVAILABLE, or UNINITIALIZED. If the action request region is BUSY then the action number, statement number, action being performed, element being acted upon and the element type are displayed.

Chapter 4: Autogen Action Option

The following information describes how a software developer can automatically generate elements that use a component that is the target of an Add, Update, or Generate action. This feature uses the Autogen action option.

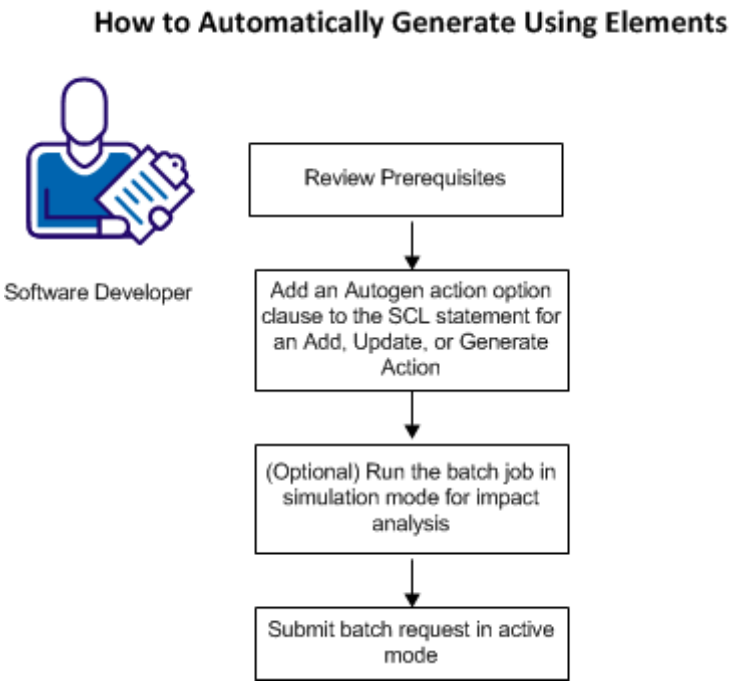
This section contains the following topics:

How to Automatically Generate "Using" Elements with Autogen (see page 51)

How to Automatically Generate "Using" Elements with Autogen

As a software developer, you can automatically generate elements that use a component. Such elements are known as *using* elements. For example, if you have a component element copybook COPYA, then the programs that use that copybook are using elements. To automatically generate using elements, you specify the Autogen action option on the Add, Update, or Generate action that will act on the component element. This option is available in batch only and cannot be used in packages.

The following graphic shows how to automatically generate using elements:



To automatically generate using elements, do the following:

1. Review Prerequisites (see page 52).
2. Code an Autogen or an Autogen Span action option in a batch processing request (see page 56).
3. (Optional) Run the Autogen batch job in simulation mode for impact analysis (see page 59).
4. Submit your batch request. For more information about submitting batch jobs, see the *User Guide*.

Note: Although the Autogen action option can only be used in batch, component elements can be generated in foreground or batch as follows:

- A Generate action generates the target of the action.
- When you add, update, transfer, or restore an element, it is automatically generated. However, the element is not generated if you set the Generate Elements option to N in foreground or specified Bypass Generate Processor in batch. These options tell CA Endevor SCM not to execute the generate processor after adding, updating, transferring or restoring the element.

Review Prerequisites

Before using the Autogen action option, complete the following prerequisites:

- Verify that the CA Endevor Automated Configuration option has been purchased and enabled for your site. You can use the Display Site panel to determine if this option is activated (ASCM=Y) at your site.
- Verify that Global Type Sequencing is enabled. This processing option causes the batch processing of element actions to occur in a Type sequence defined at the site level. Autogen requires Global Type Sequencing so that the elements that use a component are generated after all of the components have been processed (for example, macros before source, source before load modules). For more information on enabling Global Type Sequencing, see the scenario *How to Enable Global Type Sequencing*.
- Verify the AUTOGEN_SOURCE setting in the Optional Features table. When AUTOGEN_SOURCE is not turned on, Generate actions for using elements are built with the NoSource option. This is the default. 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.

- This scenario assumes you have the following knowledge:
  - How to perform batch tasks in foreground or using the ISPF panels. For more information, see Performing Batch Tasks in Foreground, in the *User Guide*.
  - How to code SCL statements. For more information, see the *SCL Reference Guide*.
- Make sure that you understand how Autogen works. Autogen generates only those using elements that are in the same logical map as the target component. Certain Autogen Span options also generate using elements that are found in different Systems and Subsystems within the Environment and Stage of the logical map. The following topics explain the different Autogen options and how they affect processing:
  - Autogen Action Option (see page 53)
  - How the Autogen Action Option Affects Processing (see page 54)
  - How Autogen Span Options Work (see page 58)
  - Autogen and Autogen Span Processing Examples (see page 59)

Autogen Action Option

When specified on an Add, Update, or Generate action for a component element, the *Autogen action option* automatically generates using elements. A *using* element is an element that uses a component element. For example, if Autogen is specified for copybook COPYA, then the programs that use that copybook are known as using elements. Specifying Autogen for an element of Type Macro automatically generates the source elements that use the macro, which then generates the appropriate LNK elements.

Autogen is available in batch only for the Add, Update, and Generate actions and cannot be used in packages. CCIDs and comments from the original generated element are used. Autogen generates *only* those using elements that are found at the same inventory location as the target component or are found higher up the logical map. To generate using elements located across Systems or Subsystems, you can use the Autogen Span options. For more information about the Span options, see Autogen Spans Across Systems and Subsystems (see page 58) in the *Administration Guide*.

If you specify the Autogen option on any one of a group of statements, then all of those statements are resolved based on the current inventory contents before any statement is executed. Statements such as GENERATE ELEMENT * create actions based on the location and options of the Generate action. During processing, duplicate Generate actions are eliminated, and the NoSource option is enabled for all the Generate actions built by Autogen. 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.

Autogen improves processing by eliminating duplicate processing of components and reduces the work that is required of users, who no longer must use the Automated Configuration Manager Query facility (ACMQ) to create additional Generate actions for element components and then run another batch job to process them.

You can run Autogen in simulation mode, to preview the effects of an Autogen request.

The following restrictions apply to Autogen:

- Autogen requires Global Type Sequencing so that the elements that use a component are generated after all of the components have been processed (for example, macros before source, source before load modules).
- 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.
- Your site must have purchased and activated the option CA Endevor Automated Configuration. You can use the Display Site panel to determine if this option is activated (ASCM=Y) at your site.
- Autogen cannot be specified on actions that are included in a package, because approvers must see the SCL statements that they are approving.
- Autogen and the bypass generate element (GENERATE ELEMENT=N) options are mutually exclusive.
- Autogen is a batch option. It cannot be specified on foreground requests.

How the Autogen Action Option Affects Processing

Autogen is available in batch only for the Add, Update, and Generate actions and cannot be used in packages. When an element is generated, either by the Generate statement or through an Add or Update request, and the Autogen option is specified, CA Endevor SCM processes the request as follows:

1. An ACMQ search is performed for all elements that use the component element being generated.
2. For each using element, a Generate action is created and added to the list of actions to be performed. However, duplicate Generate actions are eliminated, so that each using element will only be generated once, even if multiple components are being generated that are used by use the same element.

   The Generate action created for each using element will perform as follows:

   - Generate the using element in the same inventory location as the component.
   - Use the same CCID and COMMENT specified in the original statement.

- Override signout, if specified in the original statement.
- Use the NoSource option.

  Note: 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.

Accordingly, the using elements are generated as if the following Generate statement had been specified:

```
GENERATE ELEMENT name
FROM <inventory location in the original command>
TYPE type
OPTION CCID <same ccid>
COMMENT <same comment>
<override signout, if specified on original command>
NOSOURCE.
```

If the Copyback option is specified on the original action, then the Copyback only applies to the component element that is specified on the original request. The NoSource option is always used on all the Generate statements built for the using elements, unless the administrator has activated the AUTOGEN_SOURCE option in ENCOPTBL.

3. 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.
4. Global type sequencing is used to determine the order in which the elements are generated. However, with the Autogen option, SCL requests are fully resolved based on current inventory contents prior to any action being processed. When multiple actions are specified and the Autogen option is specified on at least one of the actions, all the actions are expanded before any of the actions are processed. All name masking is resolved and individual actions are built and then the expanded actions are executed according to the Global type sequence order.

   Consequently, if the Generate Autogen option is specified and several Move actions are included in the same request, the Move actions do not include any of the additional elements that are copied back as a result of the Generate action.

Example: Autogen Processing when Move actions are included on the same request

In this example, Macro MD1 exists at the DEV stage 1 location. This macro is used in program PD1 located at the DEV stage 1 location and in programs PQ2 and PQ3 located at the QA stage 2 location. The following two actions are included in the same batch job:

```
SET FROM ENV DEV SYS SYS1 SUBS SUB1 STAGE NUM 1.
GENERATE ELE MD1 OPTION AUTOGEN.
MOVE ELE *.
```

The Generate action results in elements MD1, PD1, PQ2, and PQ3 being generated at the DEV stage 1 location. However, the Move action results in only the MD1 and PD1 elements being moved to the next location in the logical map. This is because PQ2 and PQ3 were originally not at the STG1 location, so they remain at the DEV stage 1 environment.

How to Code Autogen

To automatically generate using elements, you can add one of the Autogen action options to an Add, Update, or Generate action in batch. This procedure describes how to perform this task from the foreground batch panels. As an alternative to performing batch tasks in foreground, you can perform batch tasks from the ISPF Menu.

1. Select option 3, Batch, from the Primary Options Menu.

   The Batch Options Menu opens.

2. Complete the fields as appropriate for your task. Select option 1, BUILD SCL and press Enter.

   The SCL Generate panel opens. This panel allows you to select the type of action request you want to generate,

3. Specify the option number that corresponds to the action (Add/Update or Generate) that you want to perform and then press Enter.

   After you press Enter, the action panel opens that corresponds to the action option you selected.

4. Code the Add, Update, or Generate action to include one of the following Autogen action options and verify that Bypass Generate Processor option is not set.

   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 or sourced elements, depending on the Optional Features table (ENCOPTBL) setting. To find out how the SOURCE_OPTION parameter is set in ENCOPTBL, contact your CA Endevor SCM administrator. If you specify AUTOGEN or AUTOGEN SPAN NONE in CA Endevor Quick Edit or from the Foreground Batch panel, the SCL is written as “AUTOGEN SPAN NONE”.

   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 that are 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.

   Your SCL statement for the Add/Update or Generate request include the Autogen option of your choice.

Note: Autogen, or any of the Autogen Span action options, can be specified in a 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. If you enter an 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.

How Autogen Span Options Work

Autogen generates *only* those using elements that are in the same logical map as the target component. Autogen Span options *also* generates using elements that are found in different Systems and Subsystems within the Environment and Stage of the logical map. Using elements are generated at the target Environment and Stage specified in the Autogen request, but in the System and Subsystem of the using element. The Span enhancement makes it easy to locate and generate using elements across Systems or Subsystems.

For all Autogen and Autogen Span results, any generated elements that are not currently at the target location are brought back as sourceless elements. However, 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.

Also for all Autogen and Autogen Span results, using elements are not generated if they are located in unmapped Environments or they are located lower in the map than the target location of the component.

You can run any of the Autogen Span options in simulation mode, to preview the effects of an Autogen request.

The following values are valid for the Autogen option:

- Autogen or Autogen Span None: Generates only those using elements that are in the same logical map of the component specified. Whether you specify "Autogen" or "Autogen Span None", the SCL is written as "Autogen Span None".
- Autogen Span All: Generates using elements from all Systems and Subsystems.
- Autogen Span Systems: Generates using elements from all Systems with the same-named Subsystem of the component specified.
- Autogen Span Subsystems: Generates using elements from all Subsystems with the same-named System of the component specified.

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.

Run Autogen in Simulation Mode for Impact Analysis

You can run the Autogen option in simulation mode to see what the results would be without actually performing actions with Autogen activated. Simulation mode causes the actual execution of all actions in the batch job to be bypassed. No updates are performed. This procedure assumes that you have already coded Autogen on the actions you want to process with Autogen.

Follow these steps:

1. Append the EN$AUSIM DD statement to your batch job that includes Autogen options. For Foreground Batch requests (or CA Endevor Quick Edit requests) use the Include Additional JCL option to add the DD statement. An example of the DD statement is shown next:

   ```
   //EN$AUSIM DD SYSOUT=*
   ```

2. Submit the job for processing.

   The C1MSGS1 and C1MSGS2 reports show the Syntax Request report and Action Summary report as if the actions were executed. There will be a heading line, labeled "*** AUTOGEN SIMULATION ***" to indicate that the job ran in simulation mode.

3. Review the reports for impact analysis to see what the results would be if the actions were performed.

Autogen and Autogen Span Processing Examples

The following examples show the effects of the various Autogen action options on the same elements. Each example shows the effects of one of the options on the original state of the inventory.

In all the examples, System SYSTEMC Subsystem SUB1 and SUBC contain elements that are used in compiling programs in the Application systems, SYSTEM1, SYSTEM2, and SYSTEM3. The graphics show the component element COPYC1 highlighted along with the using programs PGMC1 and PGMC2.

DEV maps to QAS, and QAS maps to PRD. The EMER environment maps to PRD. ADMIN is separate, and holds the processors that all of the other environments use.

All examples assume the following criteria:

- The Common SYSTEMC libraries are concatenated in every Generate processor for every application System in every Environment.
- Consequently, all changes made in DEV in the common Systems are automatically picked up by the application Systems at every Stage, before a package or promotion to PRD has been performed.
- For these use cases, COPYC1 is a component of all PGMC, PGMC1, and PGMC2 programs
- Each use case assumes that the elements and locations are as shown in the original chart. Use cases are not affected by prior use cases.

Before Processing Image for All Examples

Before a Generate with Autogen— The following graphic shows the elements in their original state before a Generate with Autogen request is performed.

ADMIN	ADMIN ENDEVOR - All processors & tables									
PRD	Common SYSTEMC		Application SYSTEM1		Application SYSTEM2			Application SYSTEM3		
	SUB1	SUBC	SUB1	SUB2	SUBA			SUB1	SUB2	
	-COPYC1 -PGMA1 -PGMB1 -PGMC1	-COPYA1 -PGMA2 -PGMB2 -PGMC2	-PGMA1 -PGMB1 -PGMC1 -PGMD1	-PGMA2 -PGMB2 -PGMC2 -PGMD2	-PGMA -PGMB -PGMC -PGMD			-PGMA1 -PGMB1 -PGMC1 -PGMD1	-PGMA2 -PGMB2 -PGMC2 -PGMD2	
QAS	Common SYSTEMC		Application SYSTEM1		Application SYSTEM2			Application SYSTEM3		
	SUB1	SUBC	SUB1	SUB2	SUBA			SUB1	SUB2	
DEV	Common SYSTEMC		Application SYSTEM1		Application SYSTEM2			Application SYSTEM3		
	SUB1	SUBC	SUB1	SUB2	SUB1 -PGMB -PGMC		SUB2	SUB1	SUB2	
EMER (maps to PRD)	Common SYSTEMC SUBC		Application SYSTEM1		Application SYSTEM2			Application SYSTEM3		
			SUB1	SUB2	SUBA	SUB1 -PGMB -PGMC	SUB2	SUB1	SUB2	

Example 1: Autogen or Autogen Span None

In this example, a developer changes COPYC1 in DEV Common SYSTEMC, Subsystem SUB1, and then wants to automatically generate all using elements within its same logical map. To do this, the developer performs the following Generate with Autogen or Autogen Span None request:

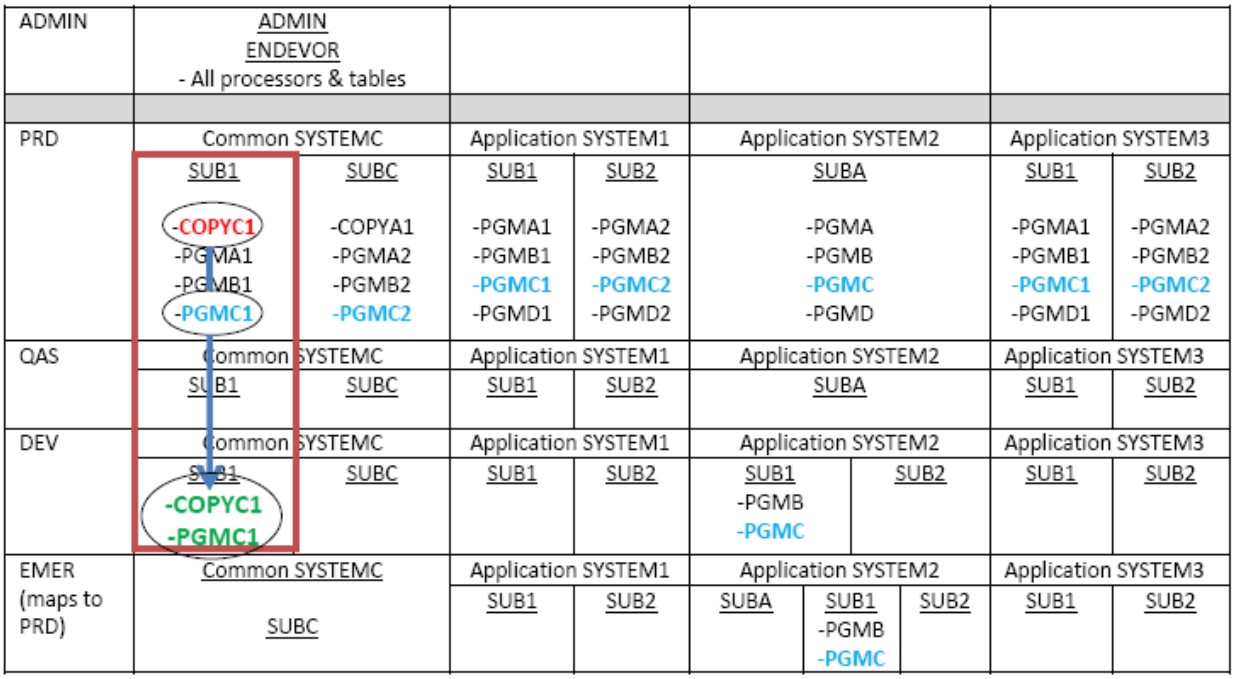
```
GENERATE ELEMENT 'COPYC1'
FROM ENVIRONMENT 'DEV'
SYSTEM 'SYSTEMC'
SUBSYSTEM 'SUB1'
TYPE COPYBOOK
STAGE NUMBER 1
OPTIONS CCID REQ#43023
COMMENT 'EDIT COMMON COPY BOOKS'
COPYBACK
AUTOGEN SPAN NONE .
```

The effects of this GENERATE request are as follows:

- Copybook COPYC1 is copied back and generated at DEV/SYSTEMC/SUB1/1.
- PGMC1 found up the map at PRD/SYSTEMC/SUB1/2 is generated at DEV/SYSTEMC/SUB1/1 as a NOSOURCE element.
- No other using programs outside the logical map of the component are affected.

Now the developer can complete his testing, and create a package with Move actions that includes all the effected elements.

After a Generate with Autogen— The following graphic shows results after a Generate with Autogen or Autogen Span None request is performed on COPYC1 at the target location Environment 'DEV', Stage 1, System 'SYSTEMC', Subsystem 'SUB1'.



Example 2: Autogen Span All

In this example, a developer changes COPYC1 in DEV Common SYSTEMC, Subsystem SUB1, and then wants to automatically generate all using elements that use the copybook in all Systems and Subsystems. To do this, the developer performs the following Generate with Autogen Span All request:

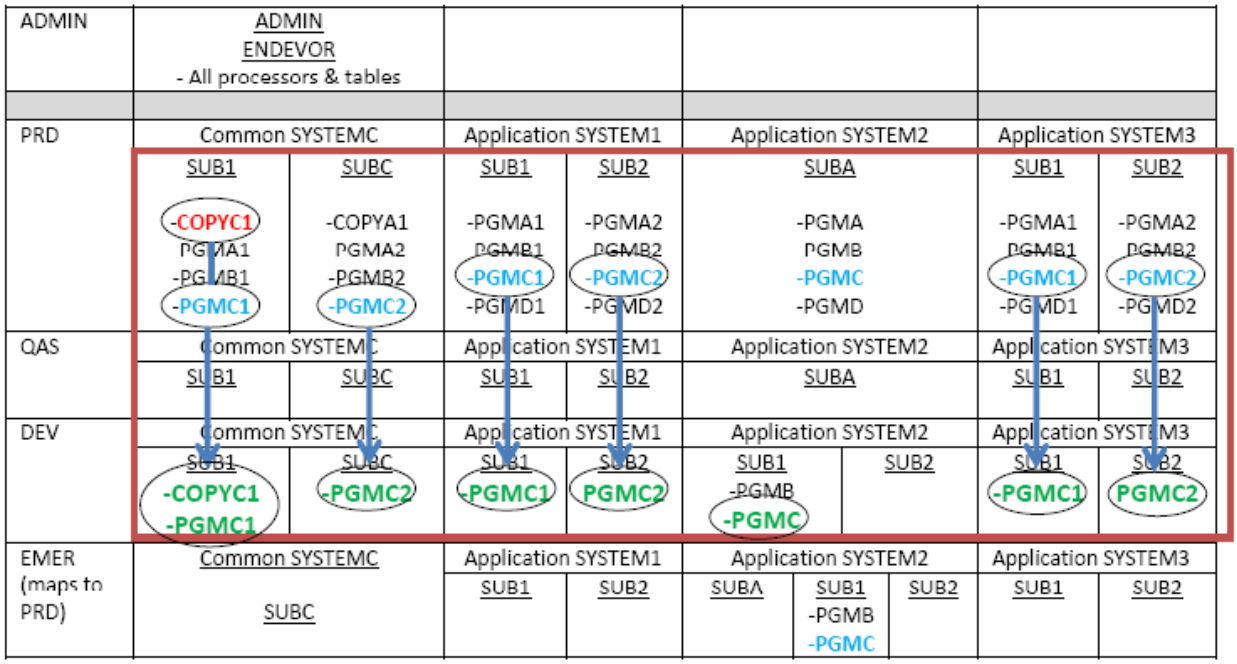
```
GENERATE ELEMENT 'COPYC1'
FROM ENVIRONMENT 'DEV'
SYSTEM 'SYSTEMC'
SUBSYSTEM 'SUB1'
TYPE COPYBOOK
STAGE NUMBER 1
OPTIONS CCID REQ#43023
COMMENT 'EDIT COMMON COPY BOOKS'
COPYBACK
AUTOGEN SPAN ALL .
```

The effects of this GENERATE request are as follows:

- Copybook COPYC1 is copied back and generated at DEV/SYSTEMC/SUB1/1.
- Element PGMC1, located up the map at PRD/SYSTEMC/SUB1/2 is generated at DEV/SYSTEMC/SUB1/1 as a NOSOURCE element.
- Element PGMC2, located outside the map at PRD/SYSTEMC/SUBC/2 is generated at DEV/SYSTEMC/SUBC/1 as a NOSOURCE element.
- Element PGMC1, located outside the map at PRD/SYSTEM1/SUB1/2 is generated at DEV/SYSTEM1/SUB1/1 as a NOSOURCE element.
- Element PGMC2, located outside the map at PRD/SYSTEM1/SUB2/2 is generated at DEV/SYSTEM1/SUB2/1 as a NOSOURCE element.
- Element PGMC, located outside the map at DEV/SYSTEM2/SUB1/1 is generated in place.
- Element PGMC, located outside the map at PRD/SYSTEM2/SUBA/2 is bypassed because Autogen Span attempts to build the generate request with PGMC's system and subsystem at the target environment, but DEV/SYSTEM2/SUBA/1 does not exist.
- Element PGMC1, located outside the map at PRD/SYSTEM3/SUB1/2 is generated at DEV/SYSTEM3/SUB1/1 as a NOSOURCE element.
- Element PGMC2, located outside the map at PRD/SYSTEM3/SUB2/2 is generated at DEV/SYSTEM3/SUB2/1 as a NOSOURCE element.
- Element PGMC, located outside the map at EMER/SYSTEM2/SUB1/2 is unaffected, since EMER is not in the environment map of component, COPYC1.

Now the developer can complete his testing, and create a package with Move actions that includes all the effected elements.

After an Autogen Span All— The following graphic shows results after a Generate with Autogen Span All request is performed on COPYC1 at the target location Environment 'DEV', Stage 1, System 'SYSTEMC', Subsystem 'SUB1'.



Example 3: Autogen Span Systems

In this example, a developer changes COPYC1 in DEV Common SYSTEMC, Subsystem SUB1, and then wants to automatically generate all using elements that use the copybook in all Systems, but not Subsystems. To do this, the developer performs the following Generate with Autogen Span Systems request:

```
GENERATE ELEMENT 'COPYC1'
FROM ENVIRONMENT 'DEV'
SYSTEM 'SYSTEMC'
SUBSYSTEM 'SUB1'
TYPE COPYBOOK
STAGE NUMBER 1
OPTIONS CCID REQ#43023
COMMENT 'EDIT COMMON COPY BOOKS'
COPYBACK
AUTOGEN SPAN SYSTEMS .
```

The effects of this GENERATE request are as follows:

- Copybook COPYC1 is copied back and generated at DEV/SYSTEMC/SUB1/1.
- Element PGMC1, located up the map at PRD/SYSTEMC/SUB1/2 is generated at DEV/SYSTEMC/SUB1/1 as a NOSOURCE element.

- Element PGMC1, located outside the map at PRD/SYSTEM1/SUB1/2 is generated at DEV/SYSTEM1/SUB1/1 as a NOSOURCE element.
- Element PGMC, located outside the map at DEV/SYSTEM2/SUB1/1 is generated in place.
- Element PGMC1, located outside the map at PRD/SYSTEM3/SUB1/2 is generated at DEV/SYSTEM3/SUB1/1 as a NOSOURCE element.
- Element PGMC, located outside the map at EMER/SYSTEM2/SUB1/2 is unaffected, since EMER is not in the environment map of component, COPYC1.
- No other Subsystems except Subsystem SUB1 are affected with an Autogen Span Systems request and thus, using elements in Subsystems SUBA, SUBC, and SUB2 are not generated.

Now the developer can complete his testing, and create a package with Move actions that includes all the effected elements.

After an Autogen Span Systems— The following graphic shows results after a Generate with Autogen Span Systems request is performed on COPYC1 at the target location Environment 'DEV', Stage 1, System 'SYSTEMC', Subsystem 'SUB1'.

ADMIN	ADMIN ENDEVOR - All processors & tables								
PRD	Common SYSTEMC		Application SYSTEM1		Application SYSTEM2			Application SYSTEM3	
	SUB1	SUBC	SUB1	SUB2	SUBA			SUB1	SUB2
	-COPYC1 -PGMA1 -PGMB1 -PGMC1	-COPYA1 -PGMA2 -PGMB2 -PGMC2	-PGMA1 -PGMB1 -PGMC1 -PGMD1	-PGMA2 -PGMB2 -PGMC2 -PGMD2	-PGMA -PGMB -PGMC -PGMD			-PGMA1 -PGMB1 -PGMC1 -PGMD1	-PGMA2 -PGMB2 -PGMC2 -PGMD2
QAS	Common SYSTEMC		Application SYSTEM1		Application SYSTEM2			Application SYSTEM3	
	SUB1	SUBC	SUB1	SUB2	SUBA			SUB1	SUB2
DEV	Common SYSTEMC		Application SYSTEM1		Application SYSTEM2			Application SYSTEM3	
	SUB1	SUBC	SUB1	SUB2	SUB1		SUB2	SUB1	SUB2
	-COPYC1 -PGMC1		-PGMC1		-PGMB -PGMC			-PGMC1	
EMER (maps to PRD)	Common SYSTEMC		Application SYSTEM1		Application SYSTEM2			Application SYSTEM3	
	SUBC		SUB1	SUB2	SUBA	SUB1 -PGMB -PGMC	SUB2	SUB1	SUB2

Example 4: Autogen Span Subsystems

In this example, a developer changes COPYC1 in DEV Common SYSTEMC, Subsystem SUB1, and then wants to automatically generate all using elements that use the copybook in all Subsystems in their current System. To do this, the developer performs the following Generate with Autogen Span Subsystems request:

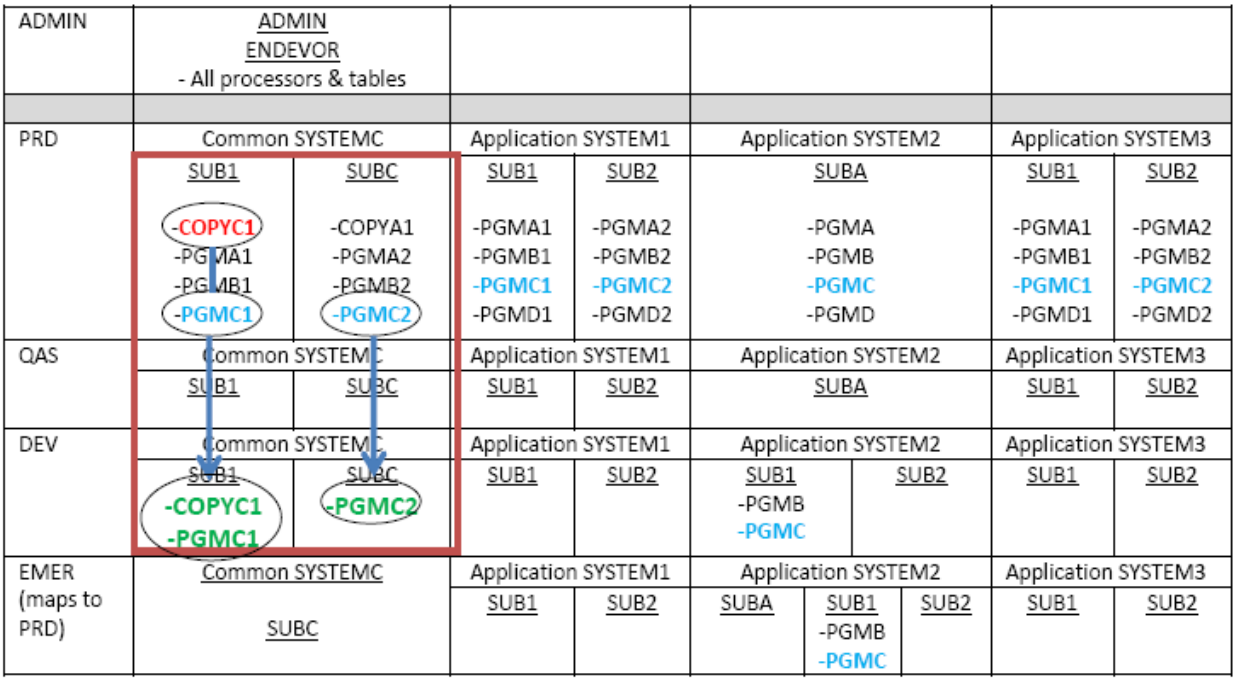
```
GENERATE ELEMENT 'COPYC1'
FROM ENVIRONMENT 'DEV'
SYSTEM 'SYSTEMC'
SUBSYSTEM 'SUB1'
TYPE COPYBOOK
STAGE NUMBER 1
OPTIONS CCID REQ#43023
COMMENT 'EDIT COMMON COPY BOOKS'
COPYBACK
AUTOGEN SPAN SUBSYSTEMS .
```

The effects of this GENERATE request are as follows:

- Copybook COPYC1 is copied back and generated at DEV/SYSTEMC/SUB1/1.
- Element PGMC1, located up the map at PRD/SYSTEMC/SUB1/2 is generated at DEV/SYSTEMC/SUB1/1 as a NOSOURCE element.
- Element PGMC2, located outside the map at PRD/SYSTEMC/SUBC/2 is generated at DEV/SYSTEMC/SUBC/1 as a NOSOURCE element.
- No other Systems are affected with an Autogen Span Subsystems request and thus, using elements in Systems SYSTEM1, SYSTEM2, and SYSTEM3 are not generated.

Now the developer can complete his testing, and create a package with Move actions that includes all the effected elements.

After an Autogen Span Subsystems— The following graphic shows results after a Generate with Autogen Span SubSystems request is performed on COPYC1 at the target location Environment 'DEV', Stage 1, System 'SYSTEMC', Subsystem 'SUB1'.



Chapter 5: Package Ship Facility

The following information describes how a CA Endevor SCM administrator can enable, use, and manage the Package Ship facility. This facility lets you to transmit package outputs (USS files, source, object, listing, or load modules), package backout members, or USS backout members from a host site to a remote site. In addition, the Post-ship Script function enables customer-written job steps (a script) to conditionally execute at the remote destination after the package ship process has completed.

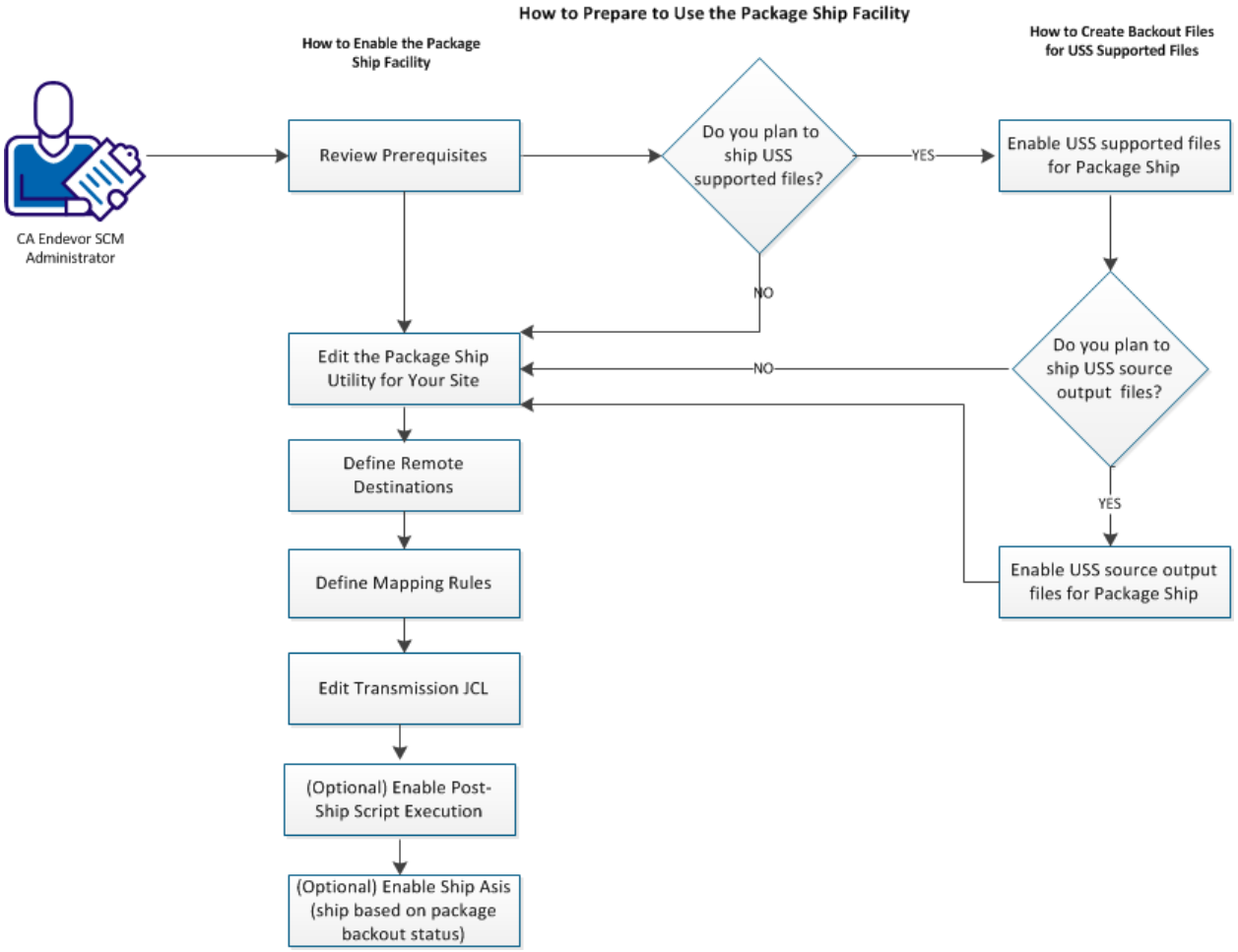
This section contains the following topics:

How to Set Up Package Ship

As a change manager (CA Endevor SCM administrator), you can enable the Package Ship facility and optionally prepare USS supported files to be shipped.

The following diagram illustrates this complex scenario that consists of the scenarios:

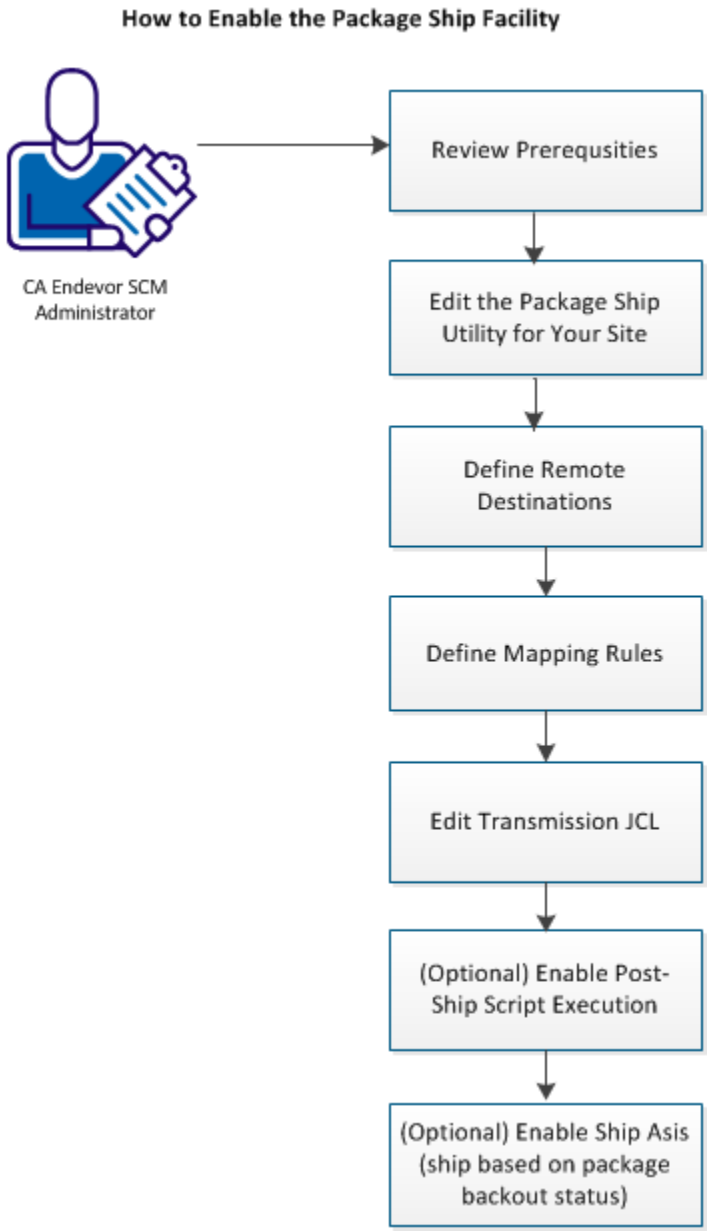
- How to Enable the Package Ship Facility (see page 71)
- How to Create Backup Files for USS Supported Files (see page 119)



How to Enable Package Ship

As a CA Endevor SCM administrator, you can enable the package ship facility to transmit package outputs (source, object, listing, or load modules), or package backout members from a host site to another site. Shipping backout members enables you to undo the prior shipment of package outputs. The utility is designed for users who develop software at a central (host) site and want to transmit source, object, or executable code to other (remote) sites.

The following diagram shows how you enable the package ship facility.



Before shipping a package, complete the following procedures to enable package ship:

1. Review Prerequisites (see page 73).
2. Configure the Package Ship Utility Program (see page 85)—The package ship utility program must be configured for you site.

   Note: If you plan to ship USS supported files (HFS, zFS, or NFS files), you must create backup files. For more information about this procedure, see How to Enable USS Supported Files for Package Ship (see page 119).
3. Establish Remote Destinations (see page 97).
4. Establish Data Set Mapping Rules (see page 102).
5. Create Model Transmission Control Statements (see page 110)—One of these data transmission packages must be available, unless the Local transmission method is used:

   - XCOM (CA)
   - Bulk Data Transfer (IBM), Version 2, or via NJE/NJI
   - NetView File Transfer Program (IBM)
   - CONNECT:Direct

   Note: Packages can also be shipped between CPUs that share DASD. This transmission method is referred to as Local.
6. (Optional) Enable the Post-ship Script feature (see page 128)— This feature enables the conditional execution of custom job steps, called scripts, at a remote destination before or after the package shipment process has completed.
7. (Optional) Enable the Ship Asis feature (see page 85)— This feature lets you ship outputs and backout members (and HFS files, or both) according to the backout status of the package. If not enabled, outputs from the time the package was executed are shipped regardless of the backout status of the package.

Review Prerequisites

Before users can ship packages, certain prerequisites are required. Complete the following prerequisites.

- Make sure that you understand how Package Ship works. Review the following topics:
  - The Package Ship Utility (see page 74)
  - How Package Ship Works (see page 75)
  - How Package Ship Works for USS Files (see page 77)
  - Ship Asis Feature (see page 79)
  - Post-Ship Script Feature (see page 80)
  - How Post-Ship Script Execution Works (see page 80)
- The Package Ship facility requires data transmission software. Verify that one of the following data transmission methods is available:
  - CA XCOM
  - Bulk Data Transfer (IBM), Version 2, or using NJE/NJI
  - NetView File Transfer Program (IBM)
  - CONNECT:Direct

  Note: Packages can also be shipped between CPUs that share DASD. This transmission method is referred to as Local.
- Verify that the following security requirements are met:
  - All IDs involved need to have OMVS segments, because these enable the IDs to start a shell environment and issue USS commands. These IDs include the IDs performing the ship (the signed on user), the Alternate ID, and the user IDs associated with the transmission method started task (XCOM, Connect direct, and so on.)
  - The transmission method started tasks must be authorized to submit jobs on behalf of the the user doing the shipment (ACID or XAUTH authority).
  - The user IDs that are executing the transmission method steps, need access to the appropriate load libraries.
  - The users involved must have read/write authority to the staging directory location prefix (by default /tmp/) and the target directory to use ACLs and or insert CHOWN CHMOD commands into the remote commands file.
  - If the optional feature PACKAGE_SHIP_WITH_ALTID is turned on, the following are required:
    - The Alternate ID needs read access to the appropriate load libraries.
    - The transmission method started tasks must be authorized to submit jobs on behalf of the Alternate ID.

The Package Ship Utility

The package shipment utility (program C1BMX000) uses data transmission programs to transmit package outputs (USS files, source, object, listing, or load modules), package backout members, or USS backout members from a host site to another site. It is designed for users who develop software at a central (host) site and want to transmit files, source, object, or executable code to other (remote) sites.

Only packages that meet the following criteria can be shipped:

- Packages must be created with the PACKAGE BACKOUT option enabled.
- Packages must have been executed, but not committed.

Note: When you use the Package Ship facility, the Ship Asis feature determines whether the backout and backin status of a package has any effect on what gets shipped. For more information, see the Ship Asis Feature (see page 79).

How Package Ship Works

When you ship a package, the request is processed at the host site and the remote site as follows.

Host site processing—When you request a package shipment, CA Endevor SCM submits a job stream at the host site. This job does the following:

1. Builds the following:
   - Staging data sets at the host site.

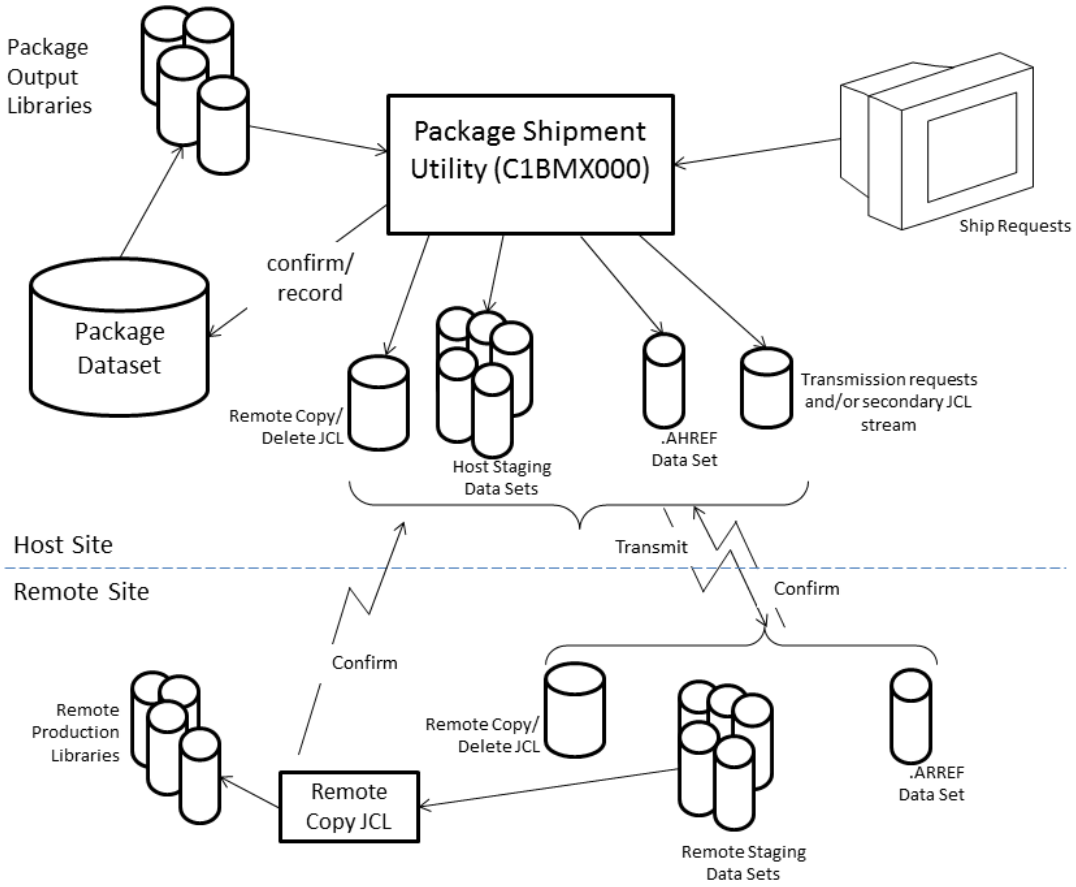
     Note: Staging data sets are dynamically created using information obtained during the Shipment execution. The High Level Qualifier of the Staging name is obtained from the MAPPING Destination, while the next level qualifiers are obtained from the DATE and TIME detail from the execution parameter. The qualifiers are created as DMMDDYY and THHMMSS. If simultaneous shipments are submitted to JES, it is possible that these qualifiers would not be unique. This case can result in JCL errors indicating that the shipment of duplicate data set names is being attempted. To avoid this failure situation, a wait mechanism should be employed by the site implementation so as to not submit shipments within the same second.
   - A job stream to copy package outputs from staging data sets to remote production data sets.
   - A file with correspondences between production and staging data sets at the host and the remote site
   - Data transmission commands to transfer all the above to the remote site. (Not applicable for Local transmission.)
2. Populates the host staging data sets with package outputs.
3. Transmits the host staging data sets and other files to the remote site. (Not applicable for Local transmission.)
4. Confirms the transmission.
5. Deletes the staging data sets at the host site (optional).

Remote site processing—When the job stream built at the host site in Step 1 executes at the remote site, it does the following:

1. Copies the package outputs from the remote staging data sets to production libraries. For Local transmission, outputs are copied from the host staging data sets to the production libraries.
2. Deletes, in the production libraries, all package members that were also deleted on the host.
3. Transmits confirmation to the host site.
4. (Optional) Deletes the staging data sets at the remote site. (Not applicable for Local transmission.)

The following diagram illustrates this basic process.

Note: Remote staging data sets are not built for Local transmission. The host staging data sets must exist on a shared DASD, and will be copied directly into the remote production libraries.



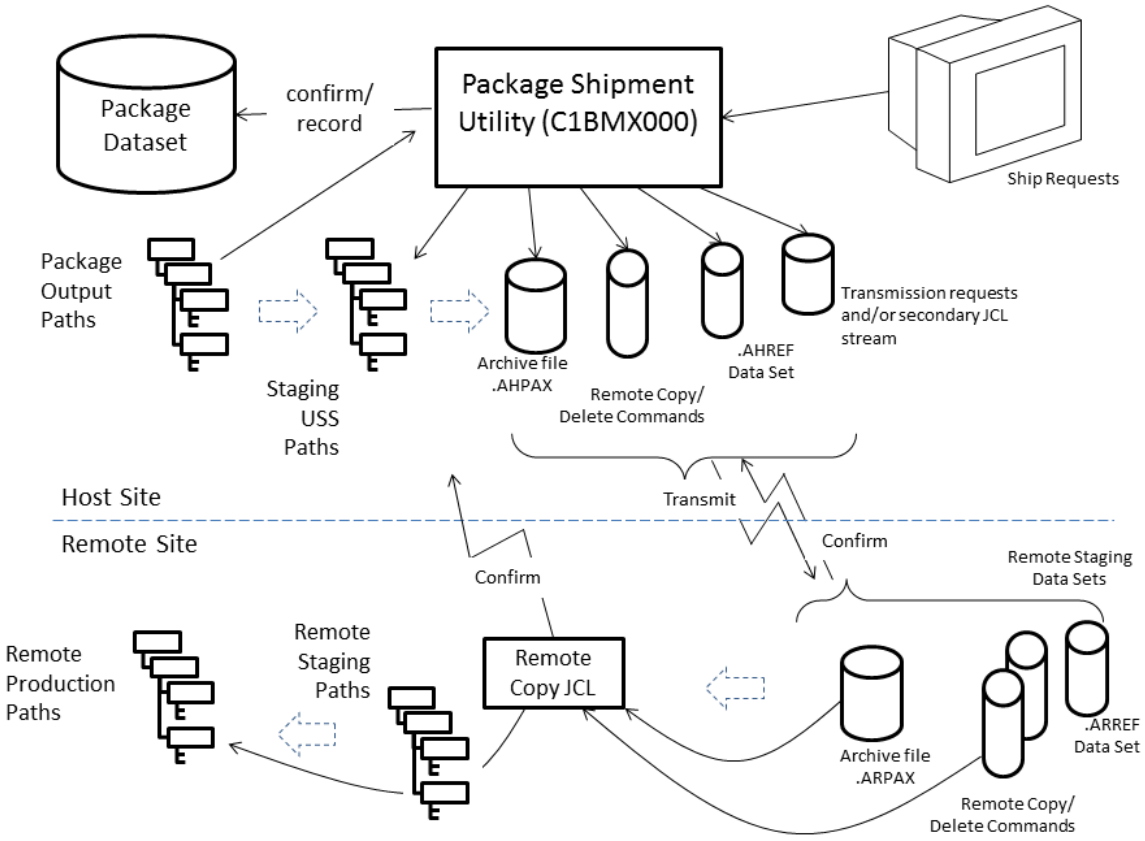
How Package Ship Works for USS Files

When you request a package that contains USS files for shipping, the process is modified slightly to include the extra steps required to ship the USS files.

1. A temporary staging path structure is allocated using the host prefix specified in the destination record.
2. The USS objects are copied into this structure, with one path for each mapped directory.

   Note: The package utility uses the backout records created by the ENUSSUTL utility to correctly identify the relevant source file at the time the package was created.
3. Optionally, if complementary files have been requested, then these backout files are also copied into their own set of paths under the staging directory.
4. The entire staging directory is compressed into an archive (.PAX) file.
5. The archive file is then transmitted to the remote location along with a job stream and commands to perform the un-archive.
6. At the remote location, the archive is expanded, and the remote copy commands (contained in the .ARUCD dataset) are executed to copy from the staging. directory into the final, production, location.
7. If complementary files were requested, then an additional file (.CRJOB) is created at the remote location containing the necessary commands to copy the complementary files from the remote staging directory if required.

The Following diagram illustrates the extensions for the USS process.


Package Dataset
confirm/ record
Package Shipment Utility (C1BMX000)
Ship Requests
Package Output Paths
Staging USS Paths
Archive file .AHPAX
Remote Copy/ Delete Commands
.AHREF Data Set
Transmission requests and/or secondary JCL stream
Host Site
Remote Site
Transmit
Confirm
Confirm
Remote Staging Data Sets
Remote Production Paths
Remote Staging Paths
Remote Copy JCL
Archive file .ARPAX
.ARREF Data Set
Remote Copy/ Delete Commands

Ship Asis Feature

When you use the Package Ship facility, the Ship Asis feature determines whether the backout and backin status of a package has any effect on what gets shipped. When enabled, this option lets you ship outputs and backout members (and HFS files, or both) according to the backout status of the package. This means that when Ship Asis is enabled and a package is partially or fully backed out, the Package Ship action has the following effects:

- Shipping the *outputs* does the following:
  - Ships the package outputs of elements that have not been backed out.
  - Ships the pre-package outputs of elements that have been backed out.
- Shipping the *backouts* does the following:
  - Ships the pre-package outputs of elements that have not been backed out.
  - Ships the package outputs of elements that have been backed out.

If Ship Asis is not enabled, outputs from the time the package was executed are shipped regardless of the backout status of the package. In this case, when you ship a package, the original outputs are shipped, even if the entire package or some of the elements have been backed out. Similarly, when you ship backouts, the original package backouts are shipped. The backout and backin status of a package has no effect on what gets shipped.

When the Ship Asis feature is enabled, the user always must take care to synchronize shipments to destination sites when the package is partially or fully backed out. With this feature enabled, during the execution of a package backout and backin action (including element action backout and backin), the user will receive messages in the package log for all destinations that this package was sent to prior to these actions. The purpose of these messages is to alert the user that the package may need to be reshipped, because these backout or backin actions have changed the package outputs. The message severity level can be set to caution or warning. For example, sample caution messages are shown next:

```
PKMR632C  PACKAGE WAS PREVIOUSLY SHIPPED TO THE FOLLOWING DESTINATION(S) PRIOR TO
[BACK-OUT| BACK-IN]
PKMR633C     JDLSHIP  19DEC12
PKMR633C     JDOSHIP  19DEC12
Reason:
This package was shipped before the backout or backin function was executed. Multiple
PKMR633C messages can appear, because one message is shown for each date and time the
package was shipped to each destination.
Action:
You may need to reship the package now that the backout function has changed the package
outputs.
```

Post-Ship Script Execution

The Post-Ship Script feature enables the conditional execution of custom job steps, called scripts, at a remote destination before or after the package shipment process has completed. Script files are created by normal processor execution and identified as "(SCRIPT-FILES)" through mapping rules. Custom script job steps are created by the Administrator by editing the shipment JCL models based on your site's requirements. In addition, a new Script Execution status can be returned and displayed in the 'Shipment Status' view by scrolling right. A script could perform a particular function, for example, a CICS New Copy or DB2 Bind.

How Post-Ship Script Execution Works

The Post-ship Script feature enables the conditional execution of custom job steps, called scripts, at a remote destination before or after the package shipment process has completed. This process works as follows:

1. The administrator creates custom script job steps by editing the shipment JCL models based on your site's requirements. These models include additional job steps in the remote job that can execute before or after the job copies the package outputs to their target data sets only if a script data set in included in the shipment. To enable conditional execution, a symbol represents each script data set. The symbol is based on the last qualifier of the host data set name, prefixed by the characters ZZ (for example, &ZZSCRIPT for the data set BST.USER12.SCRIPT). The value of the symbol is the remote staging data set name for the script data set. Therefore, the script job steps can be conditionally included in the remote job steps based on the existence of the script data set in the shipment as determined by the resolution of the data set symbol.
2. The administrator can optionally include symbols unique by destination in the script data set members. If the administrator chooses to use destination-specific symbols, the administrator will have created a destination configuration member in the PARMLIB that defines a value for the symbol that is unique to each destination. During the shipment process, symbols in the script data set members are substituted with the values from the symbols defined in the destination configuration member before being written to the host staging data set. The destinations coded in the configuration member will not be validated as defined destinations. The member will not be processed until a package shipment is executed, and only then if a script data set is included in the shipment. Messages will be produced during package ship indicating that a script data set has been found and whether or not a destination configuration member is also found and the variables substituted.
3. Your site's script data sets are created by normal processor execution.

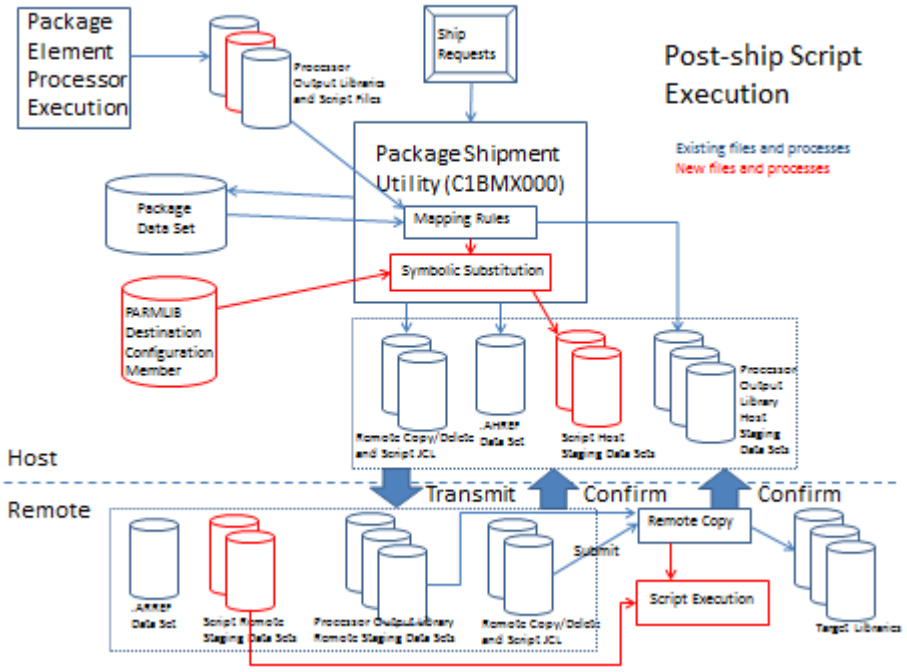
4. The script data set has to be identified as such so that the members can have their unresolved symbols resolved before the members are written to the host staging data set. During the package shipment process, the script data sets are identified as script data sets through mapping rules, where the remote data set name is defined as: (SCRIPT-FILES).

   Note: Script data sets are always shipped, but are not automatically copied to the remote target data set, unless a copy step is included in the script job steps through the model control members.

5. The host staging data set is shipped with all of the other package output data sets.

6. The remote copy job will then be executed, including the additional steps for script execution. The symbol created to represent the script data set are used by the model control members and will resolve to the remote staging data set for the script data set. This allows the script data set to be used as input for the script job steps.

The following graphic shows the package ship process including post-ship script execution:



Transmission Method Considerations

Depending upon the transmission method you use, a certain amount of information exchange and preparatory work may be necessary to transmit package outputs. This section describes the preparatory work for the following data transmission methods:

- CA XCOM
- Bulk Data transfer (BDT) using NJE/NJI
- Bulk Data transfer (BDT), Version 2
- NetView File Transfer Program (FTP)
- CONNECT:Direct
- Local file transfer

The CA XCOM Method

To define an XCOM destination to CA Endevor SCM, you must supply a remote NODENAME or an IPNAME and IPPORT address on the destination definition. The nodename is the VTAM Logical Unit name and the IPNAME and IPPORT is the TCP/IP address. The IPPORT must match the XCOM SERVPORT specification of the target server.

A destination definition can be created or updated using the Environment Option on the CA Endevor SCM Primary Options Panel in foreground or using the Define Ship Destination SCL in batch.

XCOM runs a batch job that copies data sets to a remote site and runs the job there. You need to modify the sample JCL with site-specific information. In addition, the EXEC PARM may require additional XCOM parameters.

The NDVRSHIP job step (in the Package Ship job skeleton C1BMXJOB) creates a PDS (prfx.XCOMCMD.FILE) containing a member named CONTROL. The member contains the following commands:

- A SEND TYPE=FILE command for each file to be transmitted.
- A SEND TYPE=JOB command for each destination receiving a shipment to execute the remote copy/delete job.
- A SEND TYPE=JOB command to delete the host staging data sets.

You can find the JCL and the control cards for the staging data set deletion in prfx.XCOMCMD.FILE as member "destid.D". The job step that executes XCOM (in skeleton C1BMXCOM) uses the CONTROL member as SYSIN to execute the shipment. After all transmissions are complete prfx.XCOMCMD.FILE is deleted. The XCOM Transfer Control facility (XTC) dependently synchronizes the file transmissions and job executions.

CA Endevor SCM distributes model XCOM control statements in the CSIQOPTN library as members whose names start with #PSXCOM. In general you do not need to modify these models.

Note: For more information about XCOM, see the *CA-XCOM Data Transport for z/OS User Guide*. XCOM is a product of CA.

The Bulk Data Transfer Using NJE/NJI Method

To define a BDT using a NJE/NJI destination to CA Endevor SCM, you must supply remote job information on the Destination Definition panel. This information includes the following:

- A valid JOB statement destination.
- A JES route card. The route card can be a "/*ROUTE XEQ nodename" (JES2), "//*ROUTE XEQ nodename" (JES3), or an equivalent. For more information about routing and running jobs at the remote destination, contact your JES system programmer.

CA Endevor SCM distributes model JCL in the CSIQOPTN library as members whose names start with #PSBDT1. In general, you do not need to modify these models.

The Bulk Data Transfer Version 2 Method

To define a BDT Version 2 destination to CA Endevor SCM, you must supply a remote NODENAME on the Destination Definition panel. Contact your BDT system programmer for the names of the nodes in your network.

BDT runs a batch job that copies data sets to a remote site and runs the job there. You need to modify the sample JCL with site-specific information about BDT.

Each data set you wish to transmit issues a NETREL for the remote copy/delete job. Each NETREL in turn decrements a NETHOLD equal to the number of data sets you wish to transmit. When all data sets have been transmitted, the NETHOLD is satisfied and the remote copy/delete job is released.

CA Endevor SCM distributes model BDT control statements in the CSIQOPTN library as members whose names start with #PSBDT2. In general you do not want to modify these models.

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

The Local File Transfer Method

Local shipments require that the host staging data sets be allocated on DASD which is shared between the host and remote sites. The remote production data sets can be on unshared DASD. Allocation of the shared DASD can be controlled using the UNIT and/or VOLSER fields on the Destination Definition panel.

The NetView FTP Method

NetView FTP runs a batch job that submits transactions that copy data sets to a remote site. CA Endevor SCM distributes FTP commands in the CSIQOPTN library as members whose names start with #PSNFTP. You need to modify the sample JCL with site-specific information. In general, you do not need to modify the commands.

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

The CONNECT:Direct Method

CONNECT:Direct (formerly known as Network Data Mover or NDM) runs a batch job that copies data sets from a host site to a remote destination and runs the job there. CA Endevor SCM provides sample JCL relating to CONNECT:Direct. You need to modify this sample JCL with site-specific information about CONNECT:Direct.

The NDVRSHIP jobstep (in skeleton C1BMXJOB) creates an CONNECT:Direct Process File containing a single process for each CONNECT:Direct destination. The member name of each process is "destid.P". The member contains the following commands:

- A COPY command for each file you wish to transmit.
- A RUN JOB command to execute the remote copy/delete job.
- A RUN JOB command to delete the host staging data sets for that destination.

You can find the JCL and control cards for the staging data set deletion in the process file as member(s) "destid.D". A single member "SUBMIT" contains a CONNECT:Direct SUBMIT command for each of the process members. The jobstep associated with C1BMXNDM uses the "SUBMIT" member to execute the shipment. The Process File is deleted after all processes have been submitted.

CA Endevor SCM distributes model CONNECT:Direct control statements in the CSIQOPTN library as members whose names start with #PSNWDM. In general you do not want to modify these models.

You can find additional information about CONNECT:Direct in the Product Overview. The Batch Interface is documented in the CONNECT:Direct User's Guide. Syntax for CONNECT:Direct commands can be found in the CONNECT:Direct Command Reference.

Enable Ship Asis

When the Ship Asis feature is enabled you can ship outputs and backout members (and HFS files, or both) according to the backout status of the package. During the execution of a package backout and backin action (including element action backout and backin), the user will receive warning messages in the package log. These messages list the destinations the package was shipped to before the backout function was executed. If Ship Asis is not enabled, outputs from the time the package was executed are shipped regardless of the backout status of the package.

Follow these steps:

1. Edit ENCOPTBL as follows
   a. Remove the asterisk in column one next to the option: SHIP_OUTPUTS_ASIS=ON
   b. (Optional) Remove the asterisk in column one next to the entry and set the message severity level to C (caution) or W (warning): ENHOPT MSGSEVERITY_PKMR632=
2. Compile and link ENCOPTBL.

   The Ship Asis feature is enabled and has the following effects:
   a. When a user executes a backout or backin action, messages are written to the package log that indicate the destinations to which package has already been shipped. This alerts the user to decide whether the package needs to be reshipped.
   b. When the user ships a package that has been backed out, the outputs are shipped according to the backout status of the package.

Configure the Host JCL

Modify the skeletal JCL for each step of the host execution of the package ship utility at your site. These skeletons are members in the CSIQSENU library.

- Modify the job statement JCL - C1BMXHJC
- Host Job Step 1: Shipment Staging (C1BMXJOB)
- Host Job Step 2: Transmission Scheduling
- Host Job Step 3: Confirmation JCL
- Host Job Step 4: End of Job

Host Job Statement JCL (C1BMXHJC)

Use the host job statement JCL skeleton (C1BMXHJC) to create a job statement for your host site. This job statement is embedded in the package ship job JCL (C1BMXJOB) and the job step JCL (C1BMXRCN) that confirms execution of the job that copies remote staging data sets into the remote production libraries.

The C1BMXHJC skeleton, found in the CSIQSENU library, is shown next:

```
)SEL &C1BJC1. &lnot.= &Z.
&C1BJC1.
)ENDSEL
)SEL &C1BJC2. &lnot.= &Z.
&C1BJC2.
)ENDSEL
)SEL &C1BJC3. &lnot.= &Z.
&C1BJC3.
)ENDSEL
)SEL &C1BJC4. &lnot.= &Z.
&C1BJC4.
)ENDSEL
/*ROUTE XEQ hostname
```

Modify the host job statement JCL skeleton (C1BMXHJC) as follows to provide job statement information about the host site:

- The &C1BJC.n statements expands into the JOB STATEMENT INFORMATION currently displayed on the Package Shipment panel (option **6** on the Package Options Menu). You can also replace these statements, along with their)SEL and)ENDSEL bookends with a hard-coded job statement.
- Substitute a site-specific JES Node Name value for HOSTNAME on the ROUTE statement. This causes jobs you submit at either site to be routed to the host for execution. This modification enables you to submit this job statement on both the host and the remote systems.
- Append additional JES control statements, JCL, or both as needed.
- Enable the JCL to pass security checks by including the USERID and PASSWORD keywords on the job statement, if your site requires it. You can do this by including these keywords in one of the &C1BJC.n statements. For example:

  &C1BJC2.,USERID=*userid***,PASSWORD=***password*

Host Job Step 1: Shipment Staging (C1BMXJOB)

Step 1 executes the package shipment staging utility. This is the C1BMXJOB member located in the CSIQSENU library. When modifying this skeleton, do the following:

- Remove the transmission package DD statements that do not apply (for example, if BDT is not used, remove C1BMXBDC, C1BMXBDR, and C1BMXBDM).
- Modify the remaining model control card data sets using site-specific prefixes and names. The prefix/qualifier of the CA Endevor SCM libraries is *iprfx.iqual*.

Note: The C1BMXJOB job parm contains a reference to system variable of &ZPREFIX, which corresponds to a user's TSO Profile Prefix. If users at your site typically set their profile to NOPREFIX, for example, "TSO PROFILE NOPrefix", you should change this parameter to &ZUSER or other high-level qualifier. Some of the ship JCL models use this parameter as the first node of a certain data set allocated during the ship process.

The C1BMXLIB Skeleton

This skeleton imbeds SCMM@LIB for the STEPLIB and CONLIB definitions to run the package shipment jobs at the host site.

- *C1BMXJOB*-Package shipment utility execution.
- *C1BMXHCN*-Confirm transmission of staging data sets.
- *C1BMXRCN*-Confirm remote IEBCOPY of remote staging data sets to remote production data sets.

This is the C1BMXLIB member located in the CSIQSENU library.

To modify this skeleton, modify the STEPLIB and CONLIB DD statements using a site-specific prefix, qualifier, and name. The prefix/qualifier of CA Endevor SCM libraries is iprfx.iqual. If your AUTLIBS or CONLIBS are in the linklist these two DD statements can be commented out.

The C1BMXRCN Skeleton

Use this skeleton to confirm the copying of the staging data sets to production data sets at the remote site. It is included in the C1BMXJOB skeleton.

The shipment staging utility reads this skeleton and uses it to build the part of the remote job that copies staging data sets into their respective production data sets. There are two IEBGENER steps which write JCL to the internal reader. Only one of these steps executes, based on condition codes. The job stream that is entered into the internal reader executes at the host and updates the shipment record.

No modification is necessary for this skeleton. However, you can modify it to give more specific information about the results of the copy process.

Host Job Step 2: Transmission Scheduling

Depending upon the transmission method you use, Step 2 either executes the transmission utility directly (XCOM, BDT Ver 2, and NDM), submits a job to execute the transmission utility (BDT via NJE/NJI and Netview FTP), or submits a job to copy the host staging data sets to the remote production data sets (LOCAL). The skeletons making up Step 2 may contain site-specific data set names for CA Endevor SCM and for the data transmission package you use.

C1BMXCOM JCL for XCOM

This C1BMXCOM skeleton contains four steps. The &&XXCC data set built by the package shipment utility contains IEBUPDTE SYSIN used in the two IEBUPDTE steps to populate the prfx.XCOMCMD.FILE as described in Transmission Method Considerations. XCOM is executed in the third step. The fourth step deletes the member "CONTROL" from the prfx.XCOMCMD.FILE data set.

To tailor this skeleton

1. Modify the STEPLIB and XCOMCNTL dsnames using a site-specific prefix, qualifier, and name.

   The prefix/qualifier of the XCOM libraries is iprfx.XCOM. Contact your in-house XCOM administrator to find out this information.

2. Examine the PARM and add the optional PARM(s), if desired.

C1BMXBD1 JCL for Bulk Data Transfer using NJE/NJI

Because the JCL generated for BDT through NJE/NJI varies depending upon the number of data sets you are transmitting, the actual job stream is built by the package shipment utility and written to an internal reader in this job step.

Note: For more information about the modification of this job stream (C1BMXBD1), see Creating Model Transmission Control Statements (see page 110).

C1BMXBDT JCL for Bulk Data Transfer, Version 2

Use the C1BMXBDT skeleton to transmit staging data sets and run jobs using Bulk Data Transfer, Version 2. No modification is necessary for this skeleton.

C1BMXNDM JCL for CONNECT:Direct

The C1BMXNDM skeleton contains four steps. The &&XNWC data set built by the package shipment utility contains IEBUPDTE SYSIN used in the two IEBUPDTE steps to populate the prefix.CONNCMD.FILE. CONNECT:Direct is executed in the third step. The fourth step deletes the member "SUBMIT" from the prefix.CONNCMD.FILE data set.

To customize this skeleton

1. Modify LINKLIB, NETMAP, and MSG dsnames with site-specific names. The prefix/qualifier of the CONNECT:Direct libraries is iprfx.ndm.
2. If you need a SIGNON statement, modify the SIGNON statement in the &VNBXSTP job step using site-specific signon data. You may add additional keywords and CONNECT:Direct control statements other than signon. For more information, see CONNECT: Direct Control Statements (see page 115).
3. If you do *not* need a SIGNON statement, delete the SIGNON and //SYSIN DD * statements in the &VNBXSTP job stepand code //SYSIN on the CONNCMD.FILE DD statement.

   Note: If most of your users' profiles are set to NOPREFIX, the C1BMXJOB job parm contains a reference to system variable of &VNBSHHLI, which corresponds to a user's TSO Profile Prefix. If users at your site typically set their profile to NOPREFIX, for example, "TSO PROFILE NOPrefix", you should change this parameter to &ZUSER or other high-level qualifier. Some of the ship JCL models use this parameter as the first node of a certain data sets allocated during the ship CONNCMD.

C1BMXLOC JCL for Local Transmission

Use the C1BMXLOC skeleton to transfer host staging data sets that exist on shared DASD to production data sets which may or may not exist on shared DASD. The job executes a submit command for the remote copy/delete job using batch TSO.

No modification is necessary for this skeleton.

C1BMXFTP JCL for NetView File Transfer Program

Because the Netview FTP JCL varies depending upon the number of data sets you are transmitting, the actual job stream is built by the package shipment utility and written to an internal reader in this job step. The tailoring of the C1BMXFTP job stream is discussed in Creating Model Transmission Control Statements (see page 110).

Host Job Step 3: Confirmation JCL

For transmission methods which actually execute in Step 2 (XCOM, BDT Ver 2, CONNECT:Direct, and Local), Step 3 confirms the host execution of the preceding file transmission step. It does this by executing one step if the transmission step executes, and a different step if the transmission step abends.

There is no third job step for BDT via NJE/NJI and Netview FTP. However, this skeleton is read by the first job step (DD name C1BMXHCN) and becomes part of the JCL stream which is written to the internal reader in the second job step.

Two skeletal JCL modules are involved in Step 3.

The C1BMXHCN Skeleton

To modify the C1BMXHCN skeleton, insure that the condition code on the CONFEXEC step matches the condition code in the preceding file transmission step. This skeleton can be modified to give more specific information about the results of the file transmission step.

Host Job Step 4: End of Job

The C1BMXEOJ skeleton provides support to any "end of job" processing requirements. It is a single JCL comment card that you can modify as required. The skeleton is included as the last job step of the primary JCL stream for all transmission methods. It is not included in secondary JCL streams (BDT via NJE/NJI or NetView FTP).

Modifying Remote Package Shipment JCL

CA Endevor SCM provides skeletal JCL, and model members to generate the remote Copy/Delete job of a shipment. Before using the ship utility, you must modify this skeletal JCL so that it conforms to your site conventions.

These skeletons are members in the CSIQSENU library and the model members are found in the CSIQOPTN library.

Remote Job Stream (C1DEFLTS)

The generation of the remote copy and delete job stream is controlled by the C1DEFLTS specification for RJCLROOT keyword.

RJCLROOT

Specifies the member root for the package ship remote JCL model. Controls the choice of remote JCL generation. There are three choices:

Note: The option to tailor the remote command stream, or execute Post-Ship Script processing is ONLY enabled using either RJCLROOT=ICPY or RJCLROOT=FCPY. If RJCLROOT is not specified (or omitted) these options are not available. See How to Tailor the Command Stream and Enable Post-Ship Script Execution later, for details

RJCLROOT

Not specified. Causes the job stream to be generated using IEBCOPY and IDXCAMS through a standard set of model members beginning with the character string #PSxxxx, where xxxx is specific to the transmission method being used.

RJCLROOT=FCPY

Specifies the job stream is generated through a set of model members beginning with the character string #RJFCPY, in addition to the #PSxxxx members, in order to build CA-PDSMAN FASTCOPY JCL for the remote copy and deletes.

RJCLROOT=ICPY

Specifies the job stream is generated through a set of model members beginning with the character string #RJICPY, in addition to the #PSxxxx members, in order to build the IEBCOPY and IDCAMS JCL statements.

Remote job: RJCLROOT=Not specified

If RJCLROOT is not specified, the three-step job stream is programmatically generated using IEBCOPY and IDCAMS and the #PSxxxx model control members, where xxxx is specific to the transmission method.

Remote Job Step 1: IEBCOPY JCL

This JCL copies staging data sets to their production counterparts. This job step member is read/written at the host site by the shipment staging utility and is executed at the remote site. The member is C1BMXCOP and is distributed in the CSIQSENU library.

Note: This is not an ISPF skeleton. All skeleton control statements (close parenthesis in column 1) are ignored.

Keep in mind the following:

- The utility-generated copy control cards are in IEBCOPY format. COPY and COPYMOD commands are used. No SELECT MEMBER statements are generated.
- The utility generates a //SYSIN * DD * statement.

Remote Job Step 2: IDCAMS JCL

This JCL deletes production members and remote staging data sets at the remote site. This member is read/written at the host site by the shipment staging utility and is executed at the remote site. The member is C1BMXDEL and is distributed in the CSIQSENU library.

Note: This is not an ISPF skeleton. All skeleton control statements (close parenthesis in column 1) are ignored.

```
//* *----------------------* C1BMXRJC(C1BMXDEL)
//* * REMOTE SITE JOBSTEP TO DELETE MEMBERS WHICH
//* * WERE DELETED BY THE EXECUTION OF THE PACKAGE
//* *--------------------------------*
//DELETE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
```

Remote Job Step 3: Remote confirmation

Remote confirmation steps are discussed in Host Job Step 1: Shipment Staging (C1BMXJOB), in the C1BMXRCN subsection.

Remote Job: RJCLROOT=FCPY

If RJCLROOT is equal (=) to FCPY, the job stream is generated through the set of model members beginning with the character string #RJFCPY. The distribution version consists of the following three steps:

Remote Job Step 1: Execute IEBCOPY (intercepted to execute FASTCOPY)

To copy members to and delete members from remote data sets.

Remote Job Step 2: Execute IDCAMS to delete staging datasets

Remote Job Step 3: Remote confirmation

For more information about the remote confirmation steps, see Host Job Step 1: Shipment Staging (C1BMXJOB).

Remote Job: RJCLROOT=ICPY

If RJCLROOT is equal (=) to ICPY, the job stream is generated through the set of model members in the CSIQOPTN library beginning with the character string #RJICPY. The distributed version generates the same job stream that is generated when RJCLROOT is omitted. The job stream consists of three steps:

Remote Job Step 1: Execute IEBCOPY to copy members to remote data sets

Remote Job Step 2: Execute IDCAMS to delete members and staging data sets

Remote Job Step 3: Remote confirmation

For more information about the remote confirmation steps, see Host Job Step 1: Shipment Staging (C1BMXJOB).

BPXBATCH Remote Job Step for USS Objects

If a package shipment contains any USS objects, then an extra step (BPX) is inserted into the remote copy jobs to perform the required un-archive and copy commands. As delivered, the remote commands perform the following:

trap 'exit 1;' ERR -

Enables trapping of any errors following this point.

Note: If an error is encountered, a return code 256 is set and this can be tested in the confirmation steps. For more information, see How to Build, Track and Confirm Shipments.

set -x

Enables echoing of the commands, prefixed with a plus ('+') to the STDERR output DD which makes it easier to diagnose any problems.

mkdir -p '<remoteStaginPath>'

Creates a target staging directory if it doesn't already exist.

pax -rvzk -f "//'<remoteDatasetPrefix>.ARPAX'"-s …

Performs the un-archive of the pax file.

rm -fv '<targetFile>'

Builds a corresponding delete request, for each copy, before the copy is performed. This command ensures that time stamp information is preserved in the following copy command.

cp -Bm '<stagingFile>' '<targetFile>'

Invokes a binary copy for each file maintaining the date and timestamp from the original file.

Note: The remote copy and delete command pairs will repeat for as many USS objects as need to be managed.

How to Tailor the Command Stream

You can tailor the command stream to your site standards, by inserting a step to tailor the file at the remote location. For example, you can insert mkdir, chown, or chgroup commands or modify the delivered un-pax command switches.

1. Select a model for the remote JCL. The RJCLROOT parameter in the Defaults table (C1DEFLTS) determines which custom set of models is in use at your site.
   - If your value is FCPY, or ICPY, note the value and continue to the next step.
   - If RJCLROOT is omitted or blank, then edit the C1DEFLTS to specify RJCLROOT=ICPY and reassemble the table.

   For more information about RJCLROOT, see Specifying Remote Job Stream in C1DEFLTS (see page 91).

   For more information about the Defaults table, see the chapter "Using the Defaults Table" in the *Administration Guide*.
2. Edit the copy step model JCL as follows:
   a. Locate the appropriate model in your CSIQOPTN library:
      - If your Defaults table specifies RJCLROOT=FCPY, use this model member: #RJFCPY1
      - If your Defaults table specifies RJCLROOT=ICPY use this model member: #RJICPY1

   b. Locate the BPXBATCH step in your model member. This step executes the command stream. A sample BPXBATCH step for #RJICPY1 is shown next:

```
//*
//BPXBAT   EXEC PGM=BPXBATCH           ICPY
//STDOUT   DD   SYSOUT=*
//STDERR   DD   SYSOUT=*
//STDPARM  DD   DISP=SHR,
//         DSN=&RUCD
```

      DSN=&RUCD

      Resolves the name of the remote command file.

   c. Insert steps before the BPXBATCH step to invoke your modification code according to your needs.

Example: Remote Command File Edited for REXX

Suppose that you want to have a REXX program read in the current command stream from SYSUT1 and then write the customized stream to SYSUT2. To accomplish this result, you would insert the following code before the BPXBATCH step in your model member:

```
//*************************************************************
//* SAMPLE TAILOR UCD COMMAND STREAM
//*************************************************************
//UCDEDIT EXEC PGM=IRXJCL,PARM='UCDEDIT'
//SYSEXEC  DD DISP=SHR,DSN=BST.SUPPNDVR.ISRCLIB
//SYSUT1   DD DISP=SHR,DSN=&RUCD
//SYSUT2   DD DISP=SHR,DSN=&RUCD
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//*************************************************************
```

The following sample REXX exit (UCDEDIT) illustrates the key requirements of your command stream tailoring. The program must read in the existing command stream and write out the tailored stream. The program reads the input from SUSYT1 and writes the tailored output to SYSUT2. This exit performs a simple string search and replace to amend (insert) an extra switch in the pax command.

```
/* Rexx to read from Sysut1 and write to Sysut2 (like IEBGENER)
   However, this version is a bit special in that it also performs
   some string substitutions.  It's intended to allow a customer
   to edit their USS Ship command stream at the remote location
   changing the pax command parms or saving some commands to be
   executed later.  */

frmstr = 'pax -rvzk'         /* This is what we're looking for */
tgtstr = 'pax -rvzk -p op'  /* and we'll replace it with this */

/* Read input file (BPXBATCH commands) */


"EXECIO * DISKR SYSUT1(STEM DATA. FINIS"
IF RC ¬= 0 THEN SIGNAL IOERROR



/* Main loop to search/replace the string */



 do i = 1 to DATA.0
    start = 1
    do while start < length(DATA.i)
      found = pos(frmstr,DATA.i,start)
      if found = 0 then leave
      endstr = found + length(frmstr)
      newstr = Left(DATA.i,(found - 1)) || tgtstr
      DATA.i = newstr || substr(DATA.i,endstr)
      start = found + length(tgtstr)
    end
 end


 /* Write the substituted file out */
```

```
 "EXECIO * DISKW SYSUT2(STEM DATA. FINIS"
 IF RC ¬= 0 THEN SIGNAL IOERROR



 /* That's it all done */
Exit 0



IOERROR:
    SAY 'PROBLEM DURING I/O, Return code:' || RC
EXIT RC
```

Create a Destination

You must define at least one destination for each remote site to which you plan to ship package outputs. CA Endevor SCM stores destination information in the package data set. Destinations can be defined for data sets or the USS supported file types HFS, zFS, or NFS.

Follow these steps:

1. Select option 4-Environment on the CA Endevor SCM menu.

   The Environment Options Menu.

2. Select option D- Destination on the Environment Options Menu. Then press Enter.

   The Destination Options Menu opens.

3. Select 2-Build. Type a fully qualified destination name in the Destination field. Press Enter.

   A Create/Modify Destination panel opens.

4. Complete the information on the panel. Then press Enter. This panel includes the following fields:

   DESTINATION

   Specifies a data set name or a USS supported path and file name.

   - For a data set name, the value can be a maximum of seven characters. The first character must be alphabetic. Remaining characters can be alphabetic or national.
   - For a USS system file, the value can be up to a 732-byte path name.

   DESCRIPTION

   Specifies a maximum 30-character description for the destination, which is treated as a comment.

SHIP COMPLEMENTARY DATASETS

Specifies whether to ship complementary data sets. When package outputs are shipped to a destination, a second set of staging data sets can be requested. These data sets are called complementary data sets and contain a backout of the shipment. For example, if a shipment contains the output of package A, then the complementary data sets contain the backout for package A. Conversely, if a shipment contains the backout for package B, then the complementary data sets contain the outputs for package B. Acceptable values are as follows:

- **N—**Default. Do not ship complementary data sets.
- **Y—**Ship complementary data sets.

TRANSMISSION METHOD

Specifies the transmission utility for this destination. Acceptable values are as follows:

- **XCOM—**XCOM (CA)
- **B—**Bulk Data Transfer, Version 2 (IBM)
- **BDTN—**Bulk Data Transfer via NJE/NJI (IBM)
- **L—**Local
- **NVF—**NetView File Transfer Program (IBM)
- **NWD—**CONNECT:Direct

REMOTE NODENAME

Specifies a 1-to 16-character nodename to which package outputs are to be shipped. The name must be valid for, and defined, to the chosen data transmission program. Nodename has no meaning for BDT via NJE/NJI and for local transmissions.

REMOTE IPNAME/IPPORT

Specifies the XCOM SERVPORT specification of the target server to which package outputs are to be shipped. IPname can be 1 to 63 characters. IPPORT can be 1 to 5 characters and range from 1 to 65535. The IPNAME and IPPORT specification is only valid for the XCOM transmission method.

Note: For XCOM, you must specify either the IPNAME/IPPORT or the NODENAME, but not both.

Staging Information For Data Sets

The user must provide a prefix for the staging data sets that CA Endevor SCM builds in the first step of the ship utility. This prefix can be 1 to 14 characters, and can be different for the host and the remote sites. The prefix appears in the DSN PREFIX field, and defaults to TSOuserid.NDVR.

Acceptable values for the other staging data set fields are shown in the following:

Disposition

Delete or keep. When DISPOSITION=DELETE, CA Endevor SCM includes a step in the host and/or remote JCL streams to delete the staging data sets after they have been processed. Acceptable values are:

D—Default. Delete staging data sets

K—Keep staging data sets

All staging data sets are initially cataloged. Keep/Delete refers to the final disposition of:

–Host staging data sets after they have been transmitted.

–Remote staging data sets once they have been copied to their production counterparts.

Unit

Specifies the unit designation to be used when allocating staging data sets. Default is SYSDA.

Volume Serial (Optional)

Specifies the VOLSER to be used when allocating staging data sets. There is no default value.

Staging Information For UNIX System Services files

If USS files are to be shipped for this destination, host and remote path name prefixes must be specified.

Host Path Name Prefix

Specifies a one- to 732-byte prefix for the host path name. The prefix can be different for the host site and the remote site. The default, if not specified, is "/tmp/".

Remote Path Name Prefix

Specifies a one- to 732-byte prefix for the remote path name.The prefix can be different for the host site and the remote site. The default, if not specified, is "/tmp/".

Remote Job Statement Information

These fields provide the job card for the remote JCL execution. A ROUTE XEQ card or equivalent should be included if the transmission method is either BDT via NJE/NJI or local.

The destination is created.

Modify a Destination

You can modify a destination that has already been created.

Follow these steps:

1. Select option D (DESTINATIONS) on the Environment Options Menu. Then press Enter.

   The Destination Options Menu opens.

2. Select 2-Build. Type a fully qualified destination name in the Destination field. Press Enter.

   A Create/Modify Destination panel opens. The CREATED and UPDATED fields show the date and time when the destination was created and last updated, and the user ID associated with each activity.

   Note: If you left the destination name blank or use name masking, a selection list opens. Then you can select a destination to modify from the list.

3. Change the information as necessary. Then press Enter.

   The changes you made to the destination are saved.

   Note: For more information about the fields on the Create/Modify Destination panel, see Create a Destination.

Display a Destination

You can display information for a destination that has already been defined.

Follow these steps:

1. Select option D (DESTINATIONS) on the Environment Options Menu. Then press Enter.

   The Destination Options Menu opens.

2. Select 1-Display. Type a fully qualified destination name in the Destination field. Press Enter.

   Note: If you leave the Destination field blank or use a name mask, a selection list opens. Then you must select a destination from the selection list.

   The Display Destination panel opens for the destination you specified.

Delete a Destination

You can delete a destination and any associated mapping rules that have already been created.

Follow these steps:

1. Select option D (DESTINATIONS) on the Environment Options Menu. Then press Enter.

   The Destination Options Menu opens.

2. Select 3-Delete on the Destination Options panel. Type a fully qualified destination name in the Destination field. Press Enter.

   Note: If you leave the Destination field blank or use a name mask, a selection list opens. Then you must select a destination from the selection list.

   The Delete Destination panel opens for the destination you specified.

3. Select # in the Option field, then press Enter.

   The destination and any associated mapping rules are deleted.

Note: To delete multiple destinations, select the destinations you want to delete on a Destination Selection List. When you press enter, a Delete Destination panel appears for the first selected destination. After you delete that destination, press END key. The Delete Destination panel remains displayed, with the name of the second selected destination in the DESTINATION ID field. Repeat the delete procedure for this and subsequent selected destinations.

Mapping Rules

When you create each destination, you define the staging data sets or USS directories that are built for each ship request to that destination. You must also create one or more mapping rules for each destination. A mapping rule maps production data sets or USS directories at the host to corresponding production data sets or USS directories at the remote site as follows:

- A *data set mapping rule* is a relationship between a host data set name or mask and a remote data set name or mask. Both the host and remote data set name or mask can be 1 – 44 characters in length.
- A *USS mapping rule* is a relationship between a host path name or mask and a remote path name or mask. Both the host and remote path name or mask can be up to 768 characters in length.

When creating mapping rules, consider the following factors:

- In order to match a mapping rule, CA Endevor SCM first checks that the number of qualifiers or sub-directories match. This means that the host data set name in the mapping rule must have the same number of qualifiers (separated by '.') as the input production names. For USS path names, the host path name must have the same number of sub-directories (separated by '/')as the input production path name.
- We recommend that the host names and their corresponding remote names in the mapping rule have the same number of qualifiers. This format makes the rules easier to understand and administer.

  Note: Matching begins with the highest to the lowest level qualifier. Therefore, a data set named testap.program.loadlib.clone would map to testap.program.*.* if the mapping rule options were as follows:

  ```
  testap.program.*.*
  Testap.*.loadlib.clone
  ```

  - For example, the following host data set names should be mapped to the following remote data set names:

Host data set name mask	Remote data set name mask
HOST.*.LOADLIB	REMOTE.*.LOADLIB
HOST.*.*.*	REMOTE.*.*.*

  - For example, the following host path name masks should be mapped to the following remote names:

Host path name mask	Remote path name mask
/host/*/loadlib/	/remote/*/loadlib/
/host/*/*/*/	/remote/*/*/*/

- The remote data set name fields in the mapping rules can have any number of qualifiers. If the number of qualifiers is not the same, then generally both data set names *align to the right* before building a new data set name.
  - For example, assume this mapping rule where the remote data set has more qualifiers than the host:

Host data set name mask	Remote data set name mask
TESTAP.*.*	PROD.AP.*.*

This mapping rule would result in the following remote data set being built for the following host:

Host data set	Remote data set
TESTAP.PROGRAM.LOADLIB	PROD.AP.PROGRAM.LOADLIB

  - For example, assume this mapping rule, where the host data set has more qualifiers than the remote:

Host data set name mask	Remote data set name mask
TESTAP.*.*	PROD.*

This mapping rule would result in the following remote data set being built for the following host:

Host data set	Remote data set
TESTAP.PROGRAM.LOADLIB	PROD.LOADLIB

- If the target has more nodes than the host, and the host and target start with the same explicitly defined node, but do not end with the same explicitly defined node, then the data set names *align to the left*.
  - For example, assume this mapping rule, where the remote data set has more qualifiers than the host:

Host data set name mask	Remote data set name mask
PRODAP.*.*	PRODAP.*.*.REMOTE1

This mapping rule would result in the following remote data set being built for the following host:

Host data set	Remote data set
PRODAP.A.B	PRODAP.A.B.REMOTE1

– For example, assume this mapping rule, where the remote USS path has more qualifiers than the host:

Host path name mask	Remote path name mask
/prodap/*/*/	/prodap/*/*/remote1/

This mapping rule would result in the following remote data set being built for the following host:

Host path	Remote path
/prodap/a/b/	/prodap/a/b/remote1/

- An exception to the alignment rules occurs when both the host mask and the remote mask contain no wildcards. In this case, the remote mask becomes the remote data set name or path.
- When executing a shipment request, each input data set is compared to its list of data set mapping rules until the most specific match is found. For example, the input data set TESTAP.PROGRAM.LOADLIB matches the following three host masks. The first mask, TESTAP.*.LOADLIB, is the most specific and is the one that would be used.

Input data set	Matching host data set name masks
TESTAP.PROGRAM.LOADLIB	TESTAP.*.LOADLIB
	TESTAP.*.*
	..LOADLIB

- If the remote mask is null, data sets matching the host mask are excluded from the package shipment. This behavior is useful, for example, when only load modules are shipped. This behavior enables you to set up rules to prevent the shipment of source, listings, object modules, and so on.
- If no matching rule exists, then the remote data set defaults to the same name as the input data set when executing a shipment request. If your site does not require any name changes, then all you need is a single *dummy* mapping rule, which causes every data set to map to itself. A dummy mapping rule is a mapping rule that does not include a definition for the remote data set.

  Important: If you are using a dummy mapping rule with the local shipment method, the target and source file would be the same physical data set or path. This default behavior can cause results that had not been considered during the initial planning. For example, if a new file with a USS path name has one extra node than the input masks, the data set would map to itself.

Note: The Shipment Activity Report created at run time lists the remote data set name for each host data set, with the mapping rule that was used to generate it. A DSN cross reference data set is also shipped to the remote site.

Create a Mapping Rule

A mapping rule maps production data sets or USS directories at the host to corresponding production data sets or USS directories at the remote site. Each destination requires one or more mapping rules to enable the shipment of outputs to remote sites.

Follow these steps:

1. Select option 4-Environment on the CA Endevor SCM menu.

   The Environment Options Menu.

2. Select option D (DESTINATIONS) on the Environment Options Menu. Then press Enter.

   The Destination Options Menu opens.

3. Select 2 on the Destination Options panel. Type a fully qualified destination name in the Destination field. Press Enter.

   A Create/Modify Destination panel opens.

4. Select one of the following options, then press Enter.

   C

   Create a DSN mapping rule.

   CU

   Create a USS mapping rule.

   A Create DSN Mapping Rule panel or a Create USS Mapping Rule panel opens, depending on the option you selected.

5. Complete the fields as necessary.

   Mapping rule description

   Up to 40 character description for this mapping rule.

   Fields on the Create DSN Mapping Rule panel:

   Host dataset name

   Host dataset name or mask for this mapping rule. The host dataset name must have the same number of qualifiers as the input production name.

   For script data sets, which are used for the Post-ship Script feature, the Host dataset name cannot be wildcarded. It must be explicitly specified.

   Remote dataset name

   Remote data set name or mask for this mapping rule. To exclude data sets from being transmitted, leave this field blank.

For script data sets, which are used for the Post-ship Script feature, specify the keyword exactly as shown next: (SCRIPT-FILES). The parentheses, which must be included and the dash are not valid characters for a data set name, so this value cannot be mistaken for one.

Approx host members/cyl

This field contains an approximation of the number of members that one cylinder might contain for data sets that map to this rule. The default is 16.

When packages are staged for shipment, CA Endevor SCM divides this value into the actual number of members that are being staged. Then CA Endevor SCM allocates the result +1 cylinder of primary and secondary space for the staging data sets.

Fields on the Create USS Mapping Rule panel:

Host path name

Specifies a 1- to 768- character name or mask for a path name at a host site. Host path names *must* have the same number of directories as their input production path names. To make the rules easier to understand and administer, we recommend that the host names and their corresponding remote names have the same number of qualifiers. Make sure to test your mapping rules using the Test Mapping Rules utility. For more information about testing mapping rules, see Test a Mapping Rule (see page 110).

Remote path name

Specifies a 1- to 768- character name or mask for a path at a remote site. All path names will be prefixed and appended with a forward slash (/). If you do not want USS package outputs in a host path to be transmitted, leave the remote path name blank. Remote path names can have up to 128 directories. To make the rules easier to understand and administer, we recommend that the host names and their corresponding remote names have the same number of qualifiers.Make sure to test your mapping rules using the Test Mapping Rules utility. For more information about testing mapping rules, see Test a Mapping Rule (see page 110).

Note: For more information about mapping host and remote locations, see Mapping Rules (see page 102).

Then press Enter.

The mapping rule is created.

Display a Mapping Rule

To view certain information about a mapping rule, you can display the rule. This information includes the host and the remote location to which the host maps. Who created and updated the rule and when it was created and updated are also displayed.

Follow these steps:

1. Select option D (DESTINATIONS) on the Environment Options Menu. Then press Enter.

   The Destination Options Menu opens.

2. Select 1 on the Destination Options panel. Type a fully qualified destination name in the Destination field. Press Enter.

   Note: If you leave the Destination field blank or use a name mask, a selection list opens. Then you must select a destination from the selection list.

   The Display Destination panel opens for the destination you specified.

   Note: To return to the Delete Destination panel, press End twice.

3. Select one of the following display options in the Option field on the Display Destination panel. The option you select depends on the file type that ships to the destination shown on this panel.

   D

   Displays the DSN mapping rules for the destination.

   DU

   Displays the USS mapping rules for the destination.

   The DSN Mapping Selection List panel or the USS Mapping Selection List panel opens, depending on which option you specified. The panel lists the mapping rules for the destination you specified.

4. Select the mapping rule you want to display from the selection list.

   The Display DSN Mapping Rule panel or the Display USS Mapping Rule panel opens displaying information about the mapping rule you selected from the selection list.

   - The Display DSN Mapping Rule panel displays the Host Dataset Name and the Remote Dataset Name to which it maps.
   - The Display USS Mapping Rule panel displays the Host Path Name and the Remote Path Name to which it maps.

Modify a Mapping Rule

The mode in which the Modify Mapping Rule panel displays (Create, Display, or Modify) depends on the option you select on the Destination Options panel. The exact fields that display on this panel depend on the panel mode.

Follow these steps:

1. Select option D (DESTINATIONS) on the Environment Options Menu. Then press Enter.

   The Destination Options Menu opens.

2. Select 2 on the Destination Options panel. Type a fully qualified destination name in the Destination field. Press Enter.

   A Create/Modify Destination panel opens.

3. Select one of the following options, then press Enter.

   D

   Display a DSN mapping rule for update.

   DU

   Display a USS mapping rule for update.

   The DSN Mapping Selection List panel or the USS Mapping Selection List panel opens, depending on which option you specified. The panel lists the mapping rules for the destination you specified.

4. Select the mapping rule you want to modify from the selection list.

   The Display DSN Mapping Rule panel or the Display USS Mapping Rule panel opens displaying information about the mapping rule you selected from the selection list.

   - The Display DSN Mapping Rule panel displays the Host Dataset Name and the Remote Dataset Name to which it maps.
   - The Display USS Mapping Rule panel displays the Host Path Name and the Remote Path Name to which it maps.

5. Change the fields as necessary, then press Enter.

   The mapping rule is updated.

   Note: For more information about the fields on the panel, see Create a Mapping Rule.

   Note: Press the End key until you reach the panel you need to perform your next action.

   Note: The DSN Mapping Selection List and the USS Mapping Selection List are ordered from the most specific to the least specific rule. If you change the host mask when modifying a rule, the rule is marked *RE-KEYED. This may mean that the rule has changed its relative position in the list. Redisplay the list to find the new position of the rule.

Test a Mapping Rule

You can test a mapping rule for a destination and host name. This test determines whether the remote location generates properly. The test also displays which mapping rule was used to generate the remote location.

Follow these steps:

1. Select option D (DESTINATIONS) on the Environment Options Menu. Then press Enter.

   The Destination Options Menu opens.

2. Select 2 on the Destination Options panel. Type a fully qualified destination name in the Destination field. Press Enter.

   A Create/Modify Destination panel opens.

3. Select one of the following options, then press Enter.

   T

   Test a DSN mapping rule.

   TU

   Test a USS mapping rule.

   A Test DSN Mapping Rule panel or a Test USS Mapping Rule panel for the destination opens, depending on the option you selected.

4. Type the host name, then press Enter.

   The mapping rules for this destination are searched and the remote name that corresponds to the specified host is generated and displayed on this panel. The mapping rule used to match the host to the remote location is also displayed.

Customizing Model Transmission Control Statements

To afford a measure of flexibility in transmitting data sets to different destinations, you can customize the model members that contain model transmission control statements. The number and nature of these model members depends upon the transmission package you use.

XCOM Control Statements

CA Endevor SCM distributes the XCOM control statement models in the CSIQOPTN library as a set of members whose names begin with "#PSXCOM." Those ending in a numeral direct the building of XCOM commands that schedule the transmission of data sets and the running of jobs. Those ending in a letter contain the XCOM command models. You may need to modify these command models with site-specific data or features.

When a remote site needs a different set of commands, make a copy of the #PSXCOMx member, naming it "destid.x" and make the changes. Only the members suffixed by an alpha character can be prefixed with user members, not members suffixed by a numeric character. The search order for members is "destid.x", #PSXCOMx, #PSNDVRx.

Note: For information about XCOM commands, see the XCOM User's Manual.

The XCOM control statement models in the CSIQOPTN library are described next. Modify these models as appropriate for your site.

#PSXCOME

Contains the XCOM commands to execute the remote copy/delete job and delete the host staging data sets. To prevent automatic execution at all remote sites, place an @EOF as the first statement of this member. To prevent this from happening at a particular site, create a member "destid.E" with an @EOF as the first statement.

#PSXCOMP

Contains the XCOM commands to transmit a Partitioned Data Set

#PSXCOMS

Contains the XCOM commands to transmit a Sequential Data Set

#PSXCOMT

Contains cleanup steps to delete prfx.XCOMCMD.FILE when all destinations have been processed. This member becomes the last two job steps in prfx.XCOMCMD.FILE(destid.D).

Bulk Data Transfer (BDT) Using NJE/NJI

CA Endevor SCM distributes the BDT using NJE/NJI control statements in the CSIQOPTN library as a set of members whose names begin with "#PSBDT1". Those ending in a numeral and the member suffixed with a "C" direct the building of a BDT job that does the following:

1. Unloads the partitioned staging data sets to temporary sequential data sets.
2. Executes BDTXMIT to transmit the remote site BDTRECV job.
3. Optionally deletes host staging data sets via the JCL DISP parameter.
4. Executes the transmission confirmation job step.

The remote site BDTRECV job does the following:

1. Executes BDTRECV to receive the data sets.
2. Restages the staging data sets into partitioned data sets.
3. Submits the job to copy members into the production data sets.

Those #PSBDT1 members ending in a letter contain JCL segments which may have to be modified with site-specific data. BDT via NJE/NJI also uses a set of members whose names begin with #PSNDVR. In cases where the needs of a remote site differ, make a copy of the #PSBDT1x or #PSNDVRx member, name it "destid.x" and make the changes. Only the members suffixed by an alpha character can be prefixed with user members, not members suffixed by a numeric character. The search order for members is "destid.x," #PSBDT1x, #PSNDVRx.

The BDT control statement models in the CSIQOPTN library are described next. Modify these models as appropriate for your site.

#PSBDT1E

This job step is transmitted to the remote site and is executed there after the data sets are received (as sequential files) and restaged into partitioned data sets. To prevent automatic execution at all remote sites, place an @EOF as the first statement of this member. To prevent this from happening at a particular site, create a member "destid.E" with an @EOF as the first statement.

#PSBDT1J

This job step is executed at the host site after the partitioned staging data sets have been unloaded into sequential data sets.

#PSBDT1W

This job step is executed at the remote site.

#PSNDVRH

This job step is executed at the host site and unloads the partitioned staging data sets into temporary sequential data sets prior to transmission.

#PSNDVRR

This job step is executed at the remote site and restages the received sequential data sets into temporary partitioned data sets. This control statement is set up to use the same JCL as that executed on the host. If different JCL is required, replace the @INCLUDE=H statement with the appropriate JCL.

Bulk Data Transfer (BDT), Version 2, Control Statements

CA Endevor SCM distributes the BDT Version 2 control statement in the CSIQOPTN library as a set of members whose names begin with "#PSBDT2". Those ending in a numeral direct the building of BDT commands that schedule the transmission of data sets and the running of jobs. Those ending in a letter contain the BDT command models. You may have to modify these command models with site-specific data or features.

When a remote site needs a different set of commands, make a copy of the #PSBDT2x member, naming it "destid.x" and make the changes. Only the members suffixed by an alpha character can be prefixed with user members, not members suffixed by a numeric character. The search order for members is "destid.x", #PSBDT2x, #PSNDVRx.

Note: For more information about BDT commands, see the *BDT Version 2 File-to-File Transaction Guide.*

The BDT Version 2 control statement model in the CSIQOPTN library is described next. Modify this model as appropriate for your site.

#PSBDT2E

This member contains the BDT Version 2 command to execute the remote copy/delete job. To prevent automatic execution at all remote sites, place an @EOF as the first statement of this member. To prevent this from happening at a particular site, create a member "destid.E" with an @EOF as the first statement.

Local File Transfers

Local file transfers are accomplished by submitting JCL to copy the host staging data sets directly to the remote production data sets.

CA Endevor SCM distributes Local control statements in the CSIQOPTN library as a set of members whose names begin with "#PSLOCL." In cases where the needs of a remote site differ, make a copy of the #PSLOCLx member, naming it "destid.x" and make the changes. Only the members suffixed by an alpha character can be prefixed with user members, not members suffixed by a numeric character. The search order for members is "destid.x," #PSLOCLx, #PSNDVRx.

The Local control statement model in the CSIQOPTN library is described next. Modify this model as appropriate for your site.

#PSLOCLE

This job step submits the copy/delete job for execution via TSO. To prevent automatic submission, place an @EOF as the first statement of this member. To prevent this from happening for a particular site, create a member "destid.E" with an @EOF as the first statement.

The NetView File Transfer Program

CA Endevor SCM distributes the NetView FTP control statements in the CSIQOPTN library as a set of members whose names begin with "#PSNFTP." Those ending in a numeral direct the building of FTP job which does the following:

1. Executes FTP to transmit a data set to the remote site (multiple job steps).
2. Submits the job to copy members into the production data sets.
3. Deletes the host staging data sets (optional, based on the Destination Definition).

#PSNFTP members ending in a letter contain the FTP JCL and command models and may have to be modified with site-specific data or features desired by the user. NetView FTP also uses a set of members whose names begin with #PSNDVR. In cases where the needs of a remote site differ, make a copy of the #PSNFTPx member, name it "destid.x" and make the changes. Only the members suffixed by an alpha character can be prefixed with user members, not members suffixed by a numeric character. The search order for members is "destid.x," #PSNFTPx, #PSNDVRx.

In the job stream described previously, the SUBMIT step cannot execute before the actual transmission of the data sets. This can be accomplished in one of two ways. The transmit commands can specify WAIT=YES, or the SUBMIT command of the second step can specify HOLD and be manually released.

Note: For information about FTP commands, see the appropriate NetView FTP Manual.

The NETVIEW FTP control statement models in the CSIQOPTN library are described next. Modify these models as appropriate for your site.

#PSNFTPE

This job step submits the copy/delete job for execution via TSO. To prevent automatic submission, place an @EOF as the first statement of this member. To prevent this from happening for a particular site, create a member "destid.E" with an @EOF as the first statement.

#PSNFTPJ

This job step is executed at the host site to transmit a data set.

#PSNFTPP

This member contains the FTP command which transmits a partitioned data set to the remote site. The WAIT=YES is necessary to prevent the premature execution of the "SUBMIT COPY/DELETE" step generated by #PSNFTPE.

#PSNFTPS

This member contains the FTP command which will transmit a sequential data set to the remote site. The data set name to be transmitted is located in member #PSNFTP6. The WAIT=YES is necessary to prevent the premature execution of the "submit copy/delete" step generated by #PSNFTPE.

#PSNDVRD

This job step deletes the host staging data sets if "DELETE" was specified as the Host Staging Data set Disposition.

Note: For more information about establishing destinations, see How to Establish Destinations.

CONNECT:Direct Control Statements

CA Endevor SCM distributes CONNECT:Direct control statement models in the CA Endevor SCM source library as a set of members whose names begin with "#PSNWDM". Those ending in a numeral direct the building of CONNECT:Direct commands which schedule the transmission of data sets and the running of jobs. Those ending in a letter contain the CONNECT:Direct command models and may have to be modified with site-specific data or features desired by the user.

When a remote site needs a different set of commands, make a copy of the #PSNWDMx member, name it "destid.x" and make the changes. Only the members suffixed by an alpha character can be prefixed with user members, not members suffixed by a numeric character. The search order for members is "destid.x", #PSNWDMx, #PSNDVRx.

Note: For information about CONNECT:Direct commands, see the CONNECT:Direct Command Reference.

The CONNECT:Direct control statement models in the CSIQOPTN library are described next. Modify these models as appropriate for your site.

#PSNWDMB

Contains the CONNECT:Direct Submit Statement.

#PSNWDMC

Contains the Process Statement.

#PSNWDME

Contains the CONNECT:Direct command to execute the remote copy/delete job. To prevent automatic execution at all remote sites, place an @EOF as the first statement of this member. To prevent this from happening at a particular site, create a member "destid.E" with an @EOF as the first statement.

#PSNWDMP

Contains the statements to transmit a Partitioned Data Set.

#PSNWDMS

Contains the statements to transmit Sequential Data Sets.

#PSNWDMT

Contains cleanup steps to delete prfx.CONNCMD.FILE when all destinations have been processed. It becomes the last two job steps in prfx.CONNCMD.FILE(destid.D).

How to Create External Package Shipment Job Streams

Follow this process to capture the package shipment JCL that the Package Shipment panel builds and submit it as part of a recurring data processing cycle.

1. Expand the Package Shipment JCL.
2. Move the Package Shipment JCL to a data set.
3. Formulate a method to specify which packages to ship to which destinations.

Note: If any Package Shipment skeletons change (ISPSLIB(C1BMX*)) or if the transmission method(s) to be used changes, then the external job stream must be re-constructed.

Expand the Package Shipment JCL

To expand the package shipment JCL

1. Build a shipment request for a single package to single destination.
2. On the Package Shipment panel, code TYPRUN=COPY in the HOST JOB STATEMENT INFORMATION field. Insure that the MSGCLASS points to a HELD queue.
3. Submit the shipment queue by typing **3** in the OPTION field on the Package Shipment panel. TSO displays a IKJ56250I message. Copy the jobname and job number the message displays. If no message displays, use SDSF to obtain the job name and job number.
4. Return to the Package Shipment panel and remove the TYPRUN=COPY from the HOST JOB STATEMENT INFORMATION field and restore the MSGCLASS to its original state.

   The Package Shipment JCL Stream, including all in-stream data, now resides on the JES queue.

Move the Package Shipment JCL to a Data Set

To move the Package Shipment JCL to a data set

1. Move the JES Queue to a sequential data set or to a member of a PDS. If the data set that is to contain the JCL stream does not exist, allocate it with a RECFM of "F" or "FB" and an LRECL of 80.
2. Issue the following command from the TSO READY prompt or the ISPF TSO Command Panel using the jobname and number from Step 1.

   ```
   OUTPUT jobname (jobnumber) PRINT ('data set')
   ```

   TSO displays "OUTPUT" to indicate that the JCL stream now resides in the specified data set. Press End.
3. Edit the JCL by as follows:
   a. Remove the TYPRUN=COPY from the JOB statement.
   b. Adjust any other job parameters as appropriate.
   c. Remove the JES Statistics.
   d. Ensure that the first statement in the file is the JOB statement and the last is the expansion of the ISPSLIB(C1BMXEOJ) skeleton.
   e. Modify the date and time components in *all* parameter cards to match the intended execution date and time, if necessary. These data and time fields are used to generate unique data set names and to ensure that confirmation messages are sent back for the appropriate shipment. They do not imply any scheduling restrictions. For example, in the following code, YYYYMMDD is a valid date in year, month, and date format. HHMMSSTT is a valid time (including hundredths of seconds):

      ```
      PARM='C1BMX000,YYYYMMDD,HHMMSSTT,….
      ```

   f. Tailor all the temporary data set names (in the format <prefix>.D<yymmdd>.T<hhmmss>.*) to match the date and time used in the parameter cards in the prior step. Due to limitations of the node length, a shorter version of the date and time is used. The century is omitted from the date and the hundredths of seconds from the time.

How to Specify Package and Destination Information at Run Time

Run time input to the Package Shipment Utility is read in through the C1BMXIN data set in the NDVRSHIP job step. The input is Package Shipment SCL.

Note: For more information about package shipment SCL, see Using Package Shipment SCL.

You can modify the C1BMXIN DD statement by:

- Leaving it as an in-stream data set and modifying the SCL that follows it prior to each run.
- Changing the C1BMXIN DD statement from an in-stream data set to a sequential data set or PDS member (RECFM=F or FB,LRECL=80) into which you can store the SCL until each run.
- Using SCL generators to precede the NDVRSHIP step and populate a temporary data set with SCL that will be passed through C1BMXIN.

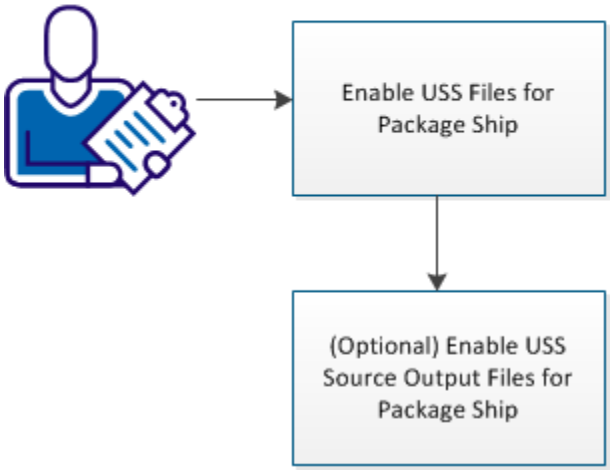
The JCL stream is now ready to be submitted.

How to Enable USS Supported Files for Package Ship

As a change manager (CA Endevor SCM administrator), you can enable USS supported files (HFS, zOS, or NFS files) for the Package Ship facility. Before USS supported files can be shipped, backout files and records must exist for these files. To create backout files and records, you add the ENUSSUTL utility to the processor associated to the element Type definition.

The following diagram shows how you enable USS files for Package Ship and, optionally, how you can enable USS source outputs for package ship.

How to Enable USS Files for Package Ship



Complete the following procedures to enable USS files for Package Ship:

1. Enable USS Files for Package Ship (see page 120).
2. (Optional) How to Enable USS Source Output Files for Package Ship (see page 124)—This process is required only If you have USS source output files that you intend to back out or ship.

More Information

The ENUSSUTL Utility (see page 126)

Enable USS Supported Files for Package Ship

Backout files and records are required to enable package outputs to be shipped using the Package Ship facility. For USS supported files, the ENUSSUTL utility creates the backout files and records. The processor that is associated to the element type must include the ENUSSUTL utility to collect and associate the backout files to the element.

Follow these steps:

1. Add the ENUSSUTL utility to the processor with the appropriate Copy and Select or Delete and Select statements.

   The ENUSSUTL syntax depends on whether it is used in a Move, Generate, or Delete processor. Move and Generate processors use Copy statements. Delete processors use Delete statements.

   - **Move and Generate processors**

     COPY syntax—Copy, Select group statements are used in Generate and Move processors. In a generate processor, copy the USS file from the CA Endevor SCM path location to a temporary path. Then use this file as input into the ENUSSUTL utility.

     ```
     Copy Indd dd-name  Outdd dd-name .
     Select File file-name [Newfile file-name].
     ```

     Copy Indd dd-name Outdd dd-name

     Identifies the dd-name copy source and target path locations.

     Indd dd-name

     Specifies the dd-name source location. CA Endevor SCM and user symbolics can be used on the JCL path name specification.

     Outdd dd-name

     Specifies the dd-name target location. CA Endevor SCM and user symbolics can be used on the JCL path name specification.

     Select FIle file-name

     Specifies the name of the file at the source location for the associated Copy statement. If the Newfile clause is not used, this file specification will be used at the target location. A file name can be up to 255 characters. See the SCL Reference Guide for information on how to specify a value that spans multiple 80 character lines. Multiple select statements can follow the copy statement. Copy and Select group statements are supported. Select statements are paired with the copy statement that precedes it. CA Endevor SCM and user symbolics can be used on the file name specification.

Newfile file-name

(Optional.) Specifies the copy file name to be used for the file at the target location. If this clause is not used, the file specification name is used. The name can be up to 255 characters. See the *SCL Reference Guide* for information on how to specify a value that spans multiple 80 character lines. CA Endevor SCM and user symbolics can be used on the file name specification.

Sample syntax for Move processors:

```
//COPY1   EXEC  PGM=ENUSSUTL
//INPUT   DD  PATH='input-pathame'
//OUTPUT  DD  PATH='output-pathname',
//        PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
//ENUSSIN   DD  *
  COPY INDD 'INPUT'  OUTDD 'OUTPUT' .
  SELECT FILE 'fileone.ext' .
  SELECT FILE 'filetwo.ext' .
```

Sample syntax for Generate processors:

```
//BPXW1   EXEC  PGM=BPXBATCH
//STDPARM DD  *
//SH cp –B 'pathname.&C1ELELMNT255.' 'temp-pathname.&C1ELELMNT255.';
//STDOUT  DD  SYSOUT=*
//STDERR  DD  SYSOUT=*
//COPY1   EXEC  PGM=ENUSSUTL
//INPUT   DD  PATH='temp-pathame'
//OUTPUT  DD  PATH='pathname',
//        PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
//ENUSSIN   DD  *
  COPY INDD 'INPUT'  OUTDD 'OUTPUT' .
  SELECT FILE &C1ELELMNT255. .
```

Rules for Copy syntax as are follows:

- Multiple SELECT statements can follow the COPY statement.
- Multiple COPY SELECT groups can be specified. Select statements are associated with the preceding Copy statement.

  ```
  COPY INDD 'INPT1'  OUTDD 'OUTP1'  .
  SELECT FILE 'fileone.ext'  .
  SELECT FILE 'filetwo.ext'  .
  COPY  INDD 'INP2'  OUTDD 'OUTP2'  .
  SELECT FILE 'filethree.ext' .
  ```

- The JCL PATHOPTS parameter is ignored if used on the Copy DD JCL statements.
- The PATHMODE parameter should be used on the OUTDD DD statement. If unspecified, the default filemode is set to '000'.

- Concatenation of paths are not allowed on the INDD DD statement. The utility will check for this and if found will fail during syntax validation.
- You cannot copy to the same path and file name.
- By default, ENUSSUTL copies the USS file using the credentials of the Alternate ID, so the owner and group of the file after the copy is the Alternate ID. If you want to use the user's credentials instead code, ALTID=N on the processor step.

- **Delete processors**

  DELETE Syntax—Use the utility's Delete and Select group statements in a delete processor.

  ```
  Delete FRomDD dd-name .
  Select File file-name .
  ```

  Delete FRomdd dd-name

  Identifies the DELETE target path locations.

  FRomdd dd-name

  Specifies the FROM dd-name. CA Endevor SCM and user symbolics can be used on the JCL path name specification.

  Select FIle file-name .

  Specifies the name of the file to be deleted that is associated with the Delete statement. A file name can be up to 255 characters. See the *SCL Reference Guide* for information on how to specify a value that spans multiple 80 character lines. Multiple select statements can follow the delete statement. Delete and Select group statements are supported. Select statements are paired with the delete statement that precedes it. CA Endevor SCM and user symbolics can be used on the file name specification.

  Sample syntax for Delete processors:

  ```
  //DEL1    EXEC  PGM=ENUSSUTL
  //FROM    DD  PATH='pathname',
  //        PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
  //ENUSSIN   DD  *
    DELETE  FROMDD 'FROM'
    SELECT FILE 'fileone.ext' .
    SELECT FILE 'filetwo.ext' .
  ```

Rules for Delete syntax as are follows:

- Multiple SELECT statements can follow the DELETE statement.
- Multiple DELETE SELECT groups can be specified. Select statements are associated with the preceding Delete statement.

```
DELETE FROMDD 'FROM1'       .
SELECT FILE 'fileone.ext'   .
SELECT FILE 'filetwo.ext'   .
DELETE FROMDD 'FROM2'       .
SELECT FILE 'filethree.ext' .
```

- The JCL PATHOPTS parameter is ignored if used on the Delete DD JCL statements.
- Concatenation of paths are not allowed on the FROMDD DD statement. The utility checks for this and if found will fail during syntax validation.
- By default, ENUSSUTL deletes the USS file using the credentials of the Alternate ID, so the Alternate ID must have appropriate access to the from path. If you want to use the user's credentials instead code, ALTID=N on the processor step.

2. (Optional) Include the NOECHO parameter on the EXEC statement, if you want to suppress the echo of ENUSSUTL commands produced by the parser from appearing in the C1MSGS1 output.

   NOECHO

   Suppresses the SCL statements from appearing in the C1MSGS1 output.

   For example, the following EXEC statement includes the NOECHO parameter and will prevent the ENUSSUTL commands from appearing in the C1MSGS1 output:

   ```
   //STEPxx  EXEC PGM=ENUSSUTL,PARM='NOECHO'
   ```

How to Enable Backout of USS Source Output Files

Source Output USS files created by CA Endevor SCM through the Type definition cannot be backed out or shipped. To have this alternate file created and to have it available for Package Backout and Ship, you must change the Type definition and Processor. One way to change the processor is to use the CONWRITE and ENUSSUTL utilities in the processor. To use this method, complete the following steps:

1. Add the CONWRITE utility to the processor to extract the element to a temporary file. For more information about CONWRITE, see the *Extended Processors Guide*.
2. Add the ENUSSUTL utility to the processor after the CONWRITE utility. Use the Copy and Select statements to specify that the ENUSSUTL utility copy the temporary file to the targeted USS file. The targeted USS file is the source output library defined on the Type definition.

   By using the ENUSSUTL utility, a backout record and a backout file is created when an action is executed under package processing control against this element. For more information about using the ENUSSUTL utility, see Enable Supported Files for Package Ship.
3. Change the Type definition to remove the SOURCE OUTPUT USS library definition.

Note: If you do not intend to back out or ship your source output USS file, you do not need to change your processor or type definition.

Example: CONWRITE and ENUSSUTL in Processor for USS Source Output Files

This example shows how a processor can use the CONWRITE and ENUSSUTL utilities to create back out files and information for USS source output files. Thus enabling the USS files to be shipped. This partial processor uses the CONWRITE utility to create USS output from a CA Endevor SCM element and copy it to a temporary USS file. Then the ENUSSUTL utility copies the USS file to its real name and location. Then the BPXBATCH utility deletes the temporary USS file.

```
//GUSS  PROC USSDIR='/u/users/endeavor/&C1EN(1,1)&C1S#/'
                      •
                      •
                      •
//*************************************************************
//* Create USS output from endevor element to a temporary USS
//* file and then use ENUSSUTL t0 copy it to its real name
//* and location.
//* Delete the temporary USS file
//* - CONWRITE element.tmp
//* - ENUSSUTL copy element.tmp to element
//* - BPXBATCH delete element.tmp
//*************************************************************
//CONW1   EXEC  PGM=CONWRITE,MAXRC=0
//ELMOUT1 DD  PATH='&USSDIR',
//        PATHOPTS=(OWRONLY,OCREAT),
//        PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
//CONWIN  DD *
 WRITE ELEMENT &C1ELMNT255
    FROM ENV &C1EN SYSTEM &C1SY  SUBSYSTEM &C1SU
         TYPE &C1TY STAGE &C1SI
    TO  DDN  ELMOUT1
        HFSFILE &C1ELMNT255..TMP
 .
//*************************************************************
//ENUSS1  EXEC  PGM=ENUSSUTL,MAXRC=4
//INPUT   DD  PATH='&USSDIR'
//OUTPUT  DD  PATH='&USSDIR',
//        PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
//ENUSSIN DD  *
 COPY INDD 'INPUT' OUTDD 'OUTPUT' .
 S FILE '&C1ELMNT255..tmp
   NEWF '&C1ELMNT255'
 .
//*************************************************************
//BPXB1  EXEC  PGM=BPXBATCH,MAXRC=0,COND=(4,LT)
//STDPARM DD  *
SH rm –r '&USSDIR.&C1ELMNT255..tmp' ;
//STDOUT  DD  SYSOUT=*
//STDERR  DD  SYSOUT=*
//*
```

The ENUSSUTL Utility

The ENUSSUTL processor utility collects package backout information for USS processor output files. This backout information consists of backout records and backout files.

Package outputs for USS files can be transmitted to other sites using the Package Ship utility. However, the output cannot be shipped unless package backout information for USS files is available. To collect this information, the processor that is associated to the element type must include the ENUSSUTL utility to collect and associate the backout files to an element. When this utility is executed under Package processing, the utility enables these files for Package Backout and Package Ship.

Generate and Move processors—The execution of the ENUSSUTL utility program in a Generate and Move processor copies user-selected USS files to the user-specified USS directory. Backout files and backout records are created for the USS files, if executed under a package.

- Backout files are stored at the target location.
- Backout records are stored in the CA Endevor SCM Package data set.

Delete processors—The execution of the ENUSSUTL utility in a Delete processor deletes user-selected USS files from a user-specified USS directory. Backout files and backout records are created for these USS files, if executed under a package.

- Backout files are stored in the same user-specified USS directory.
- Backout records are stored in the CA Endevor SCM Package data set.

How ENUSSUTL Works in a Move Processor

The ENUSSUTL utility with the Copy and Select commands in a Move processor, creates package backout records and files as follows:

1. If the select source file exists at the target location, an eight-character backout file name is generated and used as the backout saved file name. The selected file is renamed at the target location to the backout saved file name.
2. The selected Indd path and file is copied to the Outdd path and file.
3. A USS Backout record is written to the package dataset.

How ENUSSUTL Works in a Generate Processor

The ENUSSUTL utility with the Copy and Select commands in a Generate processor creates backout records and files as described by the Move processor. Before this utility is invoked in a Generate processor a step is required that copies the USS files to another temporary directory. These files can then be selected for input into the ENUSSUTL utility.

How ENUSSUTL Works in a Delete Processor

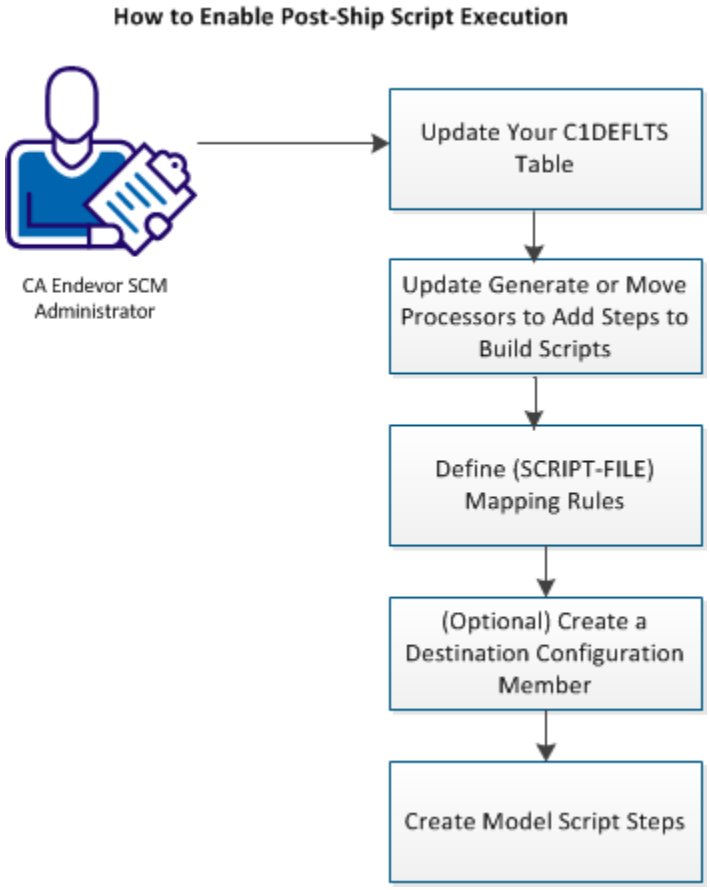
The ENUSSUTL utility with the Delete and Select commands in a Delete processor creates backout records and files as follows:

- If the select source file does not exist at the target location, no backout or rename processing is done.
- If the select source file exists at the target location, an eight-character backout file name is generated and used as the backout saved file name. The existing file is renamed to the backout saved file name in the same location.
- The selected file is deleted.
- A USS Backout record is written to the package dataset.

How to Enable Post-Ship Script Execution

As a CA Endevor SCM administrator, you can enable the Post-Ship Script feature to facilitate the conditional execution of custom job steps, called scripts, at a remote destination before or after the package shipment process has completed.

The following graphic shows how you enable the Post-Ship Script feature:



Complete the following procedures to enable Post-Ship Script execution:

1. Configure your C1DEFLTS table (see page 129). This involves reviewing your C1DEFLTS table and updating the table if necessary.
2. Update your Generate or Move Processors (see page 130) to add steps that build scripts.
3. Define mapping rules for the data sets that contain your script files (see page 131). For script data sets, you define the mapping rule's remote destination as: (SCRIPT-FILE).
4. (Optionally) Create a Destination Configuration member (see page 132) to define symbol names and values unique to for each package ship destination. This member is only required if you plan to use symbols unique by destination in the members added to a script data set.
5. Create model script steps (see page 133).

Configure C1DEFLTS for Post-Ship Script

To enable the Post-Ship Script feature, your C1DEFLTS table must meet certain requirements.

Follow these steps:

1. Review your C1DEFLTS table to confirm that your RJCLROOT parameter is set to either ICPY or FCPY. If this option was omitted, insert it with a value of RJCLROOT=ICPY. For more information on the RJCLROOT parameter, see Remote Job Stream (C1DEFLTS).
2. (Optional) Update your C1DEFLTS table as follows, if you plan to use destination script symbols:
   a. Define the name of your Destination Configuration member to C1DEFLTS, using the DESTCFGMBR= parameter. If no value is specified, the value defaults to null.
   b. Review your C1DEFLTS table to confirm that you have a valid PARMLIB dataset.
3. Reassemble your C1DEFLTS table.

Update Processors for Post-Ship Script

To enable the Post-Ship Script feature, you must update your processors to add the steps that create your script output data sets.

Follow these steps:

1. Update your generate or move processors to insert any additional steps needed to build the required remote script syntax based on your site's requirements.
2. Verify that MONITOR and BACKOUT keywords are specified on the script data set so that package processing can create backout records to track your script files.

Name your script data set so that it is easy to determine its content. The final qualifier of the script data set will be used when building model control statements. So consider choosing an appropriate name so the different types of script processing can be determined. For example, an appropriate name might be CICSNEWC for CICS Phasin requests or DB2BIND for DB2 bind requests. Any destination specific symbols that you want to be substituted at shipment **must** be present in the script file created by the processor.

Example: Create CICS New Copy Script for Post-Ship Script Execution

The following code sample will create CICS phase-in requests when the processor is executed in a package. The &CICSREGN symbol in this example will be substituted at ship time.

```
//*************************************************************
//* CREATE CICS NEW COPY SCRIPT
//*************************************************************
//PKGCHK   IF &C1PKGID NE "" THEN   * ONLY IF PACKAGE EXEC
//SCRIPCP  EXEC PGM=IEBGENER,MAXRC=0
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT2   DD DISP=SHR,MONITOR=&MONITOR,FOOTPRNT=CREATE,
//         DSN=&DSPREFIX.CICSNEWC(&C1ELEMENT)
//SYSUT1   DD DATA,DLM=##
/*VS,'F &CICSREGN,CEMT SET PRO(&C1ELEMENT),PHASEIN'
##
//PKGCHKX ENDIF
```

Define Mapping Rules for Post-Ship Script Data Sets

Define data set mapping rules for your processor output files that contain your scripts.

Follow these steps:

1. Specify the Host Dataset Name field masks to match the script data set name.
2. Specify the Remote Dataset Name field value using the keyword exactly as shown next: (SCRIPT-FILES)

   The parentheses, which must be included and the dash are not valid characters for a data set name, so this value cannot be mistaken for one.

Note: For more information about defining mapping rules, see Create a Mapping Rule (see page 106).

Example: Mapping Rule for Script Files

To create a mapping rule for a script data set with a final qualifier of CICSNEWC, define the Host Dataset Name and the Remote Dataset Name as shown next.

```
HOST DATASET NAME   ===> *.*.*.CICSNEWC
      maps to
REMOTE DATASET NAME ===> (SCRIPT-FILE)
```

Note: This example creates a mapping rule for the CICS new copy script created in the example shown in Update Processors for Post-Ship Script (see page 130).

Note: On the DSN Mapping Selection List, SCRIPT appears in the Status column when a script file is identified by a mapping rule.

Create a Destination Configuration File for Natural Objects

If you plan to use symbols that will resolve to values based on the destination in the script files created by your processors, you need to create a Destination Configuration file. The Destination Configuration member contains statements that will be read and parsed at package shipment execution time to create symbols unique to a destination that can be used in script data set members to resolve symbols for the target destination. These symbols **must** have been created in the script dataset members by your processors in order for the substitution to take place correctly.

Follow these steps:

1. Create a Destination Configuration file to define symbol names and values unique to each package ship destination.

   Use the following syntax to define the symbol names and values for each destination:

   ```
   >>--  DESTination 'destid'  -------------------------->
     +<----<----<----<----<----<----<----<----<----<---+
   --+-->  SYMbol 'symbol-name' VALue 'symbol-value' ---->
   --->  .  ------------------------------------------><
   ```

   This statement can be repeated for each destination and the SYMBOL and VALUE combination can be defined as many times as necessary within each statement. If SYMBOL is coded (there must be at least one) then VALUE must also be coded. To code a null value for a symbol, use two consecutive single quotes after the VALUE keyword.

2. Add the Destination Configuration member to your PARMLIB.

3. Include the symbols in your script data set members as appropriate to represent information that will vary by destination.

   Important: Symbol substitution can cause syntax errors by causing line overflows.

   Consider the maximum length of your symbols, use symbol substringing to generate appropriate syntax, or both. For more information on Symbol substring syntax, see the *Extended Processors Guide*. Alternatively, if for example you need to substitute long symbols such as path or file names, consider using a variable blocked (RECFM=VB) script file.

Note: The content of the Destination Configuration member can be viewed from the Display Options menu and any syntax errors can be reviewed there. The name of the Destination Configuration member is displayed in the Parameter Library Information section of the Site Information from C1DEFLTS panel. On the DSN Mapping Selection List, SCRIPT appears in the Status column when a script file is identified by a mapping rule. Script Execution status can be returned and displayed in the 'Shipment Status' view by scrolling right.

Example for Create a Destination Configuration File

Example: Destination Configuration File for Post-Ship Script

This example of a Destination Configuration file defines different values for the symbol CICSREGN depending on the target destination.

```
DESTINATION CA31XCM
    SYMBOL 'CICSREGN' VALUE 'CICS31DN'
    SYMBOL 'DB2SYS'   VALUE 'DBS31DN'  .
DESTINATION CA11LOC
    SYMBOL 'CICSREGN' VALUE 'CICS11PR'
    SYMBOL 'DB2SYS'   VALUE 'DB2S11DN'  .
```

Note: This example creates a CICSREGN symbol for the CICS new copy script created in the example shown in Update Processors for Post-Ship Script (see page 130).

Create Model Script Steps

You can write your own scripts or customize the model script job steps by editing the shipment JCL models (#RJNDVRA and #RJNDVRB in the CSIQOPTN library) based on your site's requirements. These models include job steps in the remote job, to be executed before or after the job copies the package outputs to their target data sets, based on the existence of a script data set in the shipment.

To enable conditional execution, a symbol represents each script data set. The symbol is based on the last qualifier of the host data set name, prefixed by the characters ZZ (for example, &ZZCICSNEWC for the data set BST.USER12.CICSNEWC. The value of the symbol will be the remote staging data set name for the script data set.

The symbols can be checked in the package ship model control members through @IF and @IFNOT statements. Therefore, the script job steps can be conditionally generated in the remote job steps based on the existence of the script data set in the shipment as determined by the resolution of the data set symbol.

Example: Model Post-Ship Script Steps for CICS New COPY

This model code is delivered as #RJNDVRA in the CSIQOPTN library. This code sample executes a CICS New Copy after the data set is copied to the remote target. Script files will be transmitted as PDS/Library members, one for each member that created a script, and potentially many Script files.

```
@REM ************************************************************
@REM * THESE STEPS WILL EXECUTE AFTER THE COPY TO THE REMOTE TGT
@REM ************************************************************
//* START #RJNDVRA
@IF &ZZCICSNEWC
//* START SCRIPT CICS NEW COPY
//ASCRIPT1 EXEC PGM=<YourNewCopyUtil>
//SCRIPTIN DD   DISP=SHR,
//         DSN=&ZZCICSNEWC
   ...
   <insert your site specific JCL here...>
   ...
//* END   SCRIPT 1
@ENDIF
```

Example: Model Post-Ship Script Steps to Flatten a PDS into a Sequential File

This model is delivered as #RJNDVRB in the CSIQOPTN library. This code sample flattens a PDS into a sequential file. (This example includes a modified separator line.) If your processing requires a sequential input you may need to *flatten* the library structure using your own utility or standard IBM utilities as shown in the this sample.

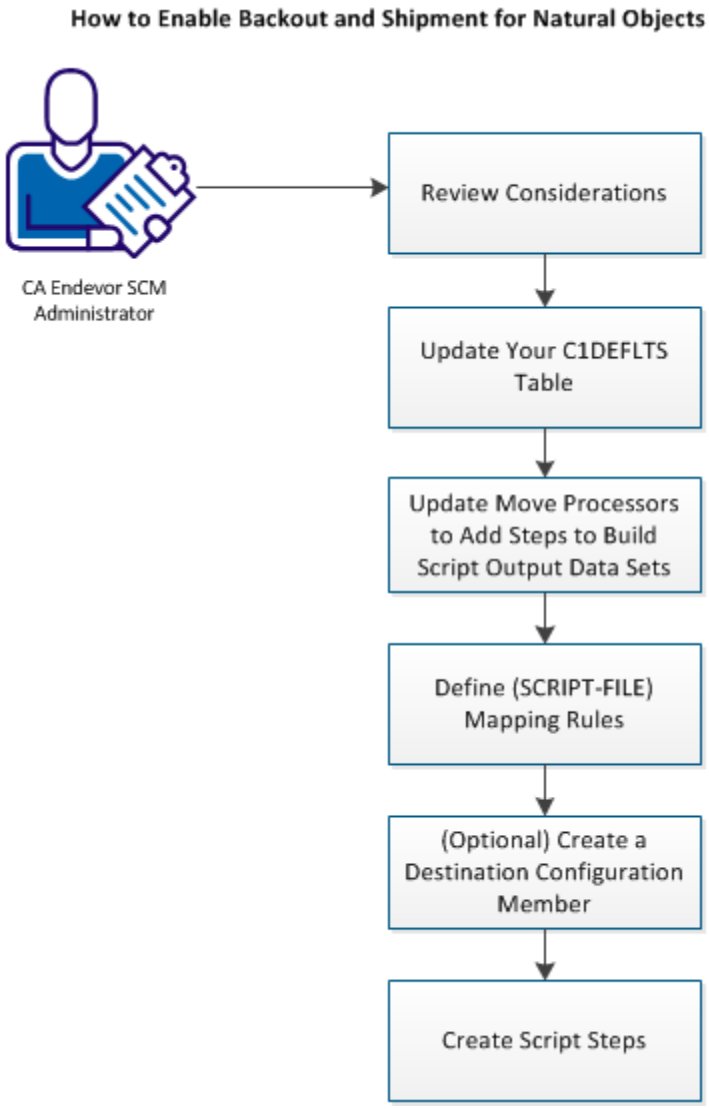
```
@REM ************************************************************
@REM * THESE STEPS provide a sample FLATTEN process
@REM ************************************************************
@IF &ZZSCRIPTFB
//*****************************************************************
//* THIS SCRIPT WILL TAKE A PDS & CREATE A FLAT (SEQUENTIAL) FILE
//* FROM ITS MEMBERS USING STANDARD IBM UTILITIES.
//*
//*****************************************************************
```

```
//*
//SCR010   EXEC PGM=IDCAMS     * FIRST CLEAN UP TEMP MEMBERS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  DEL &ZZSCRIPTFB.P
  DEL &ZZSCRIPTFB.S
  SET MAXCC = 0
//*
//SCR020   EXEC PGM=IEBPTPCH   * THEN PUNCH OUT ALL MEMBERS
//SYSPRINT DD   SYSOUT=*
//SYSUT1   DD   DISP=SHR,
//         DSN=&ZZSCRIPTFB
//SYSUT2   DD   DISP=(,CATLG),
//         DSN=&ZZSCRIPTFB.P,
//         SPACE=(CYL,(1,1),RLSE),
//         UNIT=SYSDA,
//         DCB=(RECFM=FBA,LRECL=81)
//SYSIN    DD  *
 PUNCH TYPORG=PO
//*
//SCR030   EXEC PGM=SORT       * USE SORT TO STRIP OFF LEADING
//SYSOUT   DD   SYSOUT=*       * PUNCH CONTROL CHARACTERS
//SYSPRINT DD   SYSOUT=*       * AND REPLACE MEMBER CARDS
//SYSUDUMP DD   SYSOUT=*
//REPORT1  DD   SYSOUT=*
//SORTIN   DD   DISP=SHR,
//         DSN=&ZZSCRIPTFB.P
//SORTOUT  DD   DSN=&ZZSCRIPTFB.S,
//         DISP=(,CATLG),
//         UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),
//         DCB=(RECFM=FB,LRECL=80)
//SORTWK01 DD  UNIT=DISK,SPACE=(CYL,(5,5))
//SORTWK02 DD  UNIT=DISK,SPACE=(CYL,(5,5))
//SORTWK03 DD  UNIT=DISK,SPACE=(CYL,(5,5))
//SYSIN    DD  *
  SORT FIELDS=COPY
  INREC  IFTHEN=(WHEN=(2,13,CH,EQ,C'MEMBER NAME  '),
           BUILD=(C'./ ADD NAME=',15,8)),
         IFTHEN=(WHEN=NONE,BUILD=(1:2,80))
//*
//* END SCRIPT 1
@ENDIF
@IFNOT &ZZSCRIPTFB
//* START SCRIPT - THIS STEP WILL RUN IF THERE IS NO SCRIPT FILE
//BSCRIPT1 EXEC PGM=IEFBR14,COND=((0,LE),ONLY) NO SCRIPT STEP
//* END SCRIPT 1
@ENDIF
```

How to Enable Backout and Shipment for Natural Objects

As a CA Endevor SCM administrator, you can enable the shipment of Natural objects to a remote site and the backout of those objects from the remote site. This functionality is enabled by the Post-Ship Script feature, which lets you ship job steps (scripts) to the remote destination for conditional execution. In this case, the scripts are needed to write the Natural objects to the remote Natural library after shipment.

The following graphic shows how you enable backout and shipment for Natural objects:



Complete the following procedures to enable backout and shipment of Natural Objects:

1. Review the considerations for the backout and shipment of Natural objects (see page 138).
2. Configure your C1DEFLTS table (see page 139) to enable the Post-Ship Script feature. This involves reviewing your C1DEFLTS table and updating the table if the feature is not already enabled.
3. Update your Move processors (see page 140) to add steps that will build your script output data sets for shipment and create backout records to enable backout of these data sets. The model processor steps enable package processing for shipment and backout, store unloaded Natural objects into PDS format, and put remote Natload JCL into the NATCMD library.
4. Define mapping rules for the data sets that contain your script files (see page 143). For these script data sets, you always define the mapping rule remote destination as: (SCRIPT-FILE).
5. (Optionally) Create a Destination Configuration member (see page 132) to define unique symbol values for each package ship destination. This member is only required if you want the symbols in your script data sets to resolve to values that are based on the destination.
6. Create script steps (see page 145) that can be conditionally executed at the remote destination, based on the existence of a script data set in the shipment. The model script creates a sequential file from the NATCMD file PDS members being shipped and then loads the shipped Natural objects to their Natural libraries.

Backout and Shipment of Natural Objects

The shipment of Natural objects to a remote site and the backout of those objects from the remote site is enabled by the Post-Ship Script feature. This feature lets you ship job steps (scripts) to the remote destination for conditional execution. The scripts write the Natural objects to the remote Natural library after shipment.

For Natural objects, backout and shipment has the following limitations:

- The Natural objects that you plan to ship must have valid PDS names. The Move processor for Natural requires steps that write members to two PDSs, using the CA Endevor SCM element name as the member name. Therefore, all objects being shipped must have names that will be valid as member names of a PDS.

- Each package that you plan to ship can contain a maximum of about 250 elements. This limitation is due to a Natural utility requirement that sequential files be used. At the remote site, the script file is submitted to the internal reader as a single job that consists of sequential job steps created from the PDS members. This job uses the same job card as all other jobs executed at the remote location. Thus, the number of job steps limits the shipments to approximately 250 elements per package.

  To avoid the 250 element per package limitation, you can make each member its own job, by including the job card in the JCL written to the script file member.

- Delete of Natural objects is not supported. The backout of shipped objects is possible by shipping complementary files (a destination option):
  - Submission of the CRJOB file at the remote location (CHJOB for LOCAL shipments) will restore pre-existing Natural objects to their state prior to shipment.
  - Deletes of new objects at remote locations as part of backout processing is not handled by this process.

Configure C1DEFLTS for Post-Ship Script

To enable the Post-Ship Script feature (required for backout and shipment of Natural objects), your C1DEFLTS table must meet certain requirements.

Follow these steps:

1. Review your C1DEFLTS table to confirm that your RJCLROOT parameter is set to either ICPY or FCPY. If this option was omitted, insert it with a value of RJCLROOT=ICPY. For more information on the RJCLROOT parameter, see Remote Job Stream (C1DEFLTS).
2. (Optional) Update your C1DEFLTS table as follows, if you plan to use destination script symbols:
   a. Define the name of your Destination Configuration member to C1DEFLTS, using the DESTCFGMBR= parameter. If no value is specified, the value defaults to null.
   b. Review your C1DEFLTS table to confirm that you have a valid PARMLIB dataset.
3. Reassemble your C1DEFLTS table.

Update Processors for Post-Ship Script Processing of Natural Objects

To enable the Post-Ship Script feature, you must update your Move processor to add the steps that create your script output data sets for shipment and create backout records to enable backout of these data sets.

Follow these steps:

1. Update your Move processor to insert any additional steps needed to build the required remote script Natural LOAD JCL syntax based on your site's requirements. For Natural objects, the additional steps for the Move processor must meet the following requirements:
   - Write members to two PDSs, using the CA Endevor SCM element name as the member name. This requires that all objects being shipped have names that will be valid as member names of a PDS.
     - The first PDS members contain the individual unload of the Natural objects. This is not a script file.
     - The second PDS members contain a complete job step to write the Natural object to the remote Natural library after shipment. This data set must be identified as a (SCRIPT-FILE) during shipment. This is done through the mapping rules based on the shipment destination. For more information about the SCRIPT-FILE, see Define Mapping Rules for Post-Ship Script Data Sets (see page 143).
   - Because each Natural object is individually unloaded, each must also be individually loaded back to Natural.
   - Because the script file members contain a complete job step, symbols can be used that will be substituted at shipment time based on the destination, through the destination configuration member of the CA Endevor SCM PARMLIB.
     - You must be careful not to use existing processor symbols; &C1 symbol names are not allowed in the destination configuration member.
     - Symbols, to be resolved at shipment time, can be used for the values: Natural nucleus name, parameters to the nucleus, and the remote target Natural library name.
2. Update the CA Endevor SCM PARMLIB destination configuration member, if symbols are used in the script file members.
   - Each destination that will receive the shipment must be updated to include the symbols used in the script file members.
   - Each destination can have its own value for the symbols.
3. Verify that MONITOR and BACKOUT keywords are specified on the script data set so that package processing can create backout records to track your script files.

4. Name your script data set so that it is easy to determine its content. The final qualifier of the script data set will be used when building model control statements. So consider choosing an appropriate name so the different types of script processing can be determined. For example, an appropriate name might be NATCMD for the script file containing the Natural LOAD JCL steps.

Example: Model Code to Create Post-Ship Script Output Data Sets for Natural Objects

The following code sample added to the PNATMOV Move processor for Natural objects will enable package processing for shipment and backout, store unloaded Natural objects into PDS format, and put remote Natload JCL into the NATCMD library. This sample uses symbols that will be resolved at shipment time based upon shipment destination. These symbols all begin with &RMT.

```
// OBJMT='&C1ELMNT255',
// OBJMD='&C1ELMNT255',
// OBJMS='NAT&C1ELMNT255(1,4)',
// OBJWN='WITH,NEWLIBRARY,&RMTLNAME,%',
// OBJLT='LOAD &C1ELMNT255,LIB,*,OBJTYPE,N,&OBJWN',
// OBJLD='LOAD &C1ELMNT255,LIB,*,OBJTYPE,D,%',
// OBJLS='LOAD &C1ELMNT255,LIB,*,OBJTYPE,E,&OBJWN',
// OBJMEM=&OBJM&C1TY(3,1)
//*
//*********************************************************************
//* PACKAGE PROCESSING FOR SHIPMENT AND BACKOUT (NO DDMS)
//*********************************************************************
//IF1      IF   (&C1PKGID NE ' ') THEN
//IF2      IF   (&C1TY  NE 'NTDDM') THEN
//*
//*********************************************************************
//* STORE UNLOADED NATURAL OBJECT INTO PDS
//*********************************************************************
//GENER1   EXEC PGM=IEBGENER,MAXRC=0
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
//SYSUT1   DD   DSN=&&ELMMOVE,DISP=OLD
//SYSUT2   DD   DISP=SHR,
//         DSN=BST.MOTM.ESCM715.NATOBJ(&OBJMEM),
//         MONITOR=COMPONENTS
//*
```

```
//*********************************************************************
//* PUT REMOTE NATLOAD JCL INTO NATCMD LIBRARY
//*********************************************************************
//GENER2   EXEC PGM=IEBGENER,MAXRC=0
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
//SYSUT1   DD   DATA,DLM='$$'
//*
//NATLOAD  EXEC PGM=&RMTNATNUC
//STEPLIB  DD   DSN=&RMTNATLIB,DISP=SHR
//         DD   DSN=&RMTADALIB,DISP=SHR
//CMPRMIN  DD   *
&RMTPRM1
&RMTPRM3
&RMTPRM4
&RMTPRM5
//DDCARD   DD   *
&RMTDDCARD
//CMWKF01  DD   DISP=SHR,
//         DSN=PUBLIC.MORMI08.NATOBJ(&OBJMEM)
//CMPRINT  DD   SYSOUT=*
//DDPRINT  DD   SYSOUT=*
//DDDRUCK  DD   DUMMY
//MPMDUMP  DD   DUMMY
//DDKARTE  DD   DUMMY
//CMSYNIN  DD   *
SYSPROF
SYSOBJH
&OBJL&C1TY(3,1)
WHERE,REPLACE,ALL,REPORT
.
FIN
$$
//SYSUT2   DD   DISP=SHR,
//         DSN=BST.MOTM.ESCM715.NATCMD(&OBJMEM),
//         MONITOR=COMPONENTS
//*
//IF2      ENDIF
//IF1      ENDIF
```

Define Mapping Rules for Post-Ship Script Data Sets for Natural Objects

Define data set mapping rules for your processor output files that contain your scripts.

Follow these steps:

1. Specify the Host Dataset Name field masks to match the script data set name.
2. Specify the Remote Dataset Name field value using the keyword exactly as shown next: (SCRIPT-FILE)

   The parentheses, which must be included and the dash are not valid characters for a data set name, so this value cannot be mistaken for one.

Note: For more information about defining mapping rules, see Create a Mapping Rule (see page 106).

Example: Mapping Rule for the NATCMD Script File

To create a mapping rule for a script data set with a final qualifier of NATCMD , define the Host Dataset Name and the Remote Dataset Name as shown next.

```
HOST DATASET NAME   ===> *.*.*.NATCMD
      maps to
REMOTE DATASET NAME ===> (SCRIPT-FILE)
```

Note: This example creates a mapping rule for the Natural LOAD JCL created in the example shown in Update Processors for Post-Ship Script (see page 130) Processing of Natural Objects

Note: On the DSN Mapping Selection List, SCRIPT appears in the Status column when a script file is identified by a mapping rule.

Create a Destination Configuration File for Natural Objects

If you plan to use symbols that will resolve to values based on the destination in the script files created by your processors, you need to create a Destination Configuration file. The Destination Configuration member contains statements that will be read and parsed at package shipment execution time to create symbols unique to a destination that can be used in script data set members to resolve symbols for the target destination. These symbols **must** have been created in the script dataset members by your processors in order for the substitution to take place correctly.

Follow these steps:

1. Create a Destination Configuration file to define symbol names and values unique to each package ship destination.

   Use the following syntax to define the symbol names and values for each destination:

   ```
   >>--  DESTination 'destid'  -------------------------->
     +<----<----<----<----<----<----<----<----<----<---+
   --+-->  SYMbol 'symbol-name' VALue 'symbol-value' ---->
   --->  .  ------------------------------------------><
   ```

   This statement can be repeated for each destination and the SYMBOL and VALUE combination can be defined as many times as necessary within each statement. If SYMBOL is coded (there must be at least one) then VALUE must also be coded. To code a null value for a symbol, use two consecutive single quotes after the VALUE keyword.

2. Add the Destination Configuration member to your PARMLIB.

3. Include the symbols in your script data set members as appropriate to represent information that will vary by destination.

   Important: Symbol substitution can cause syntax errors by causing line overflows.

   Consider the maximum length of your symbols, use symbol substringing to generate appropriate syntax, or both. For more information on Symbol substring syntax, see the *Extended Processors Guide*. Alternatively, if for example you need to substitute long symbols such as path or file names, consider using a variable blocked (RECFM=VB) script file.

Note: The content of the Destination Configuration member can be viewed from the Display Options menu and any syntax errors can be reviewed there. The name of the Destination Configuration member is displayed in the Parameter Library Information section of the Site Information from C1DEFLTS panel. On the DSN Mapping Selection List, SCRIPT appears in the Status column when a script file is identified by a mapping rule. Script Execution status can be returned and displayed in the 'Shipment Status' view by scrolling right.

Example: Destination Configuration File for Post-Ship Script

This example of a Destination Configuration file defines different values for the symbol RMTNATNUC and RMTNATLIB, depending on the target destination.

```
DESTINATION CA31XCM
    SYMBOL 'RMTNATNUC' VALUE 'N41DB99B'
    SYMBOL 'RMTNATLIB' VALUE 'BST.NAT414.LOAD'  .
DESTINATION CA11LOC
    SYMBOL 'RMTNATNUC' VALUE 'N42DB99B'
    SYMBOL 'RMTNATLIB' VALUE 'BST.NAT427.LOAD'  .
```

Note: This example creates &RMTNATNUC and &RMTNATLIB symbols for the Natural LOAD JCL script created in the example shown in Update Processors for Post-Ship Script Processing of Natural Objects (see page 140).

Create Script Steps for Natural Objects

You can write your own scripts or customize the model script job steps by editing the shipment JCL models (#RJNDVRA and #RJNDVRB in the CSIQOPTN library) based on your site's requirements. These models include job steps in the remote job, to be executed before or after the job copies the package outputs to their target data sets, based on the existence of a script data set in the shipment.

To enable conditional execution, a symbol represents each script data set. The symbol is based on the last qualifier of the host data set name, prefixed by the characters ZZ (for example, &ZZNATCMD for the data set BST.USER12.NATCMD. The value of the symbol will be the remote staging data set name for the script data set.

The symbols can be checked in the package ship model control members through @IF and @IFNOT statements. Therefore, the script job steps can be conditionally generated in the remote job steps based on the existence of the script data set in the shipment as determined by the resolution of the data set symbol.

Example: Model Post-Ship Script Steps for Natural Objects

The following script is provided in the model control member #RJNDVRA. This script will create a sequential file from the NATCMD file PDS members being shipped and then loads the shipped Natural objects to their Natural libraries.

The model member #RJNDVRA includes the following steps for processing the script file:

- Using standard IBM utilities (IEBPTPCH, SORT) to sequentialize all shipped members in the script file
- Submitting the sequentialized job steps created from the members to the internal reader as a single job
  - This job uses the same job card as all other jobs executed at the remote location.
  - This will limit shipments to approximately 250 elements per package (number of job steps limitation).
  - It is also possible to include the job card in the JCL written to the script file member, removing the 250 element per package limitation by making each member its own job.

Note: To use the model #RJNDVRA script, you do not need to make any edits to the #RJNDVRB model.

```
@IF &ZZNATCMD

//*********************************************************************
//*
//*
//* THESE STEPS CREATE A SEQUENTIAL FILE FROM THE NATCMD FILE PDS
//* MEMBERS BEING SHIPPED AND THEN
//* LOADS THE SHIPPED NATURAL OBJECTS TO THEIR
//* NATURAL LIBRARIES.
//*
//*********************************************************************
//* DELETE WORK FILES
//*********************************************************************
//SCR010   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  DEL PUBLIC.MORMI08.NATCMDT
  DEL PUBLIC.MORMI08.NATCMDS
  SET MAXCC = 0
//*
```

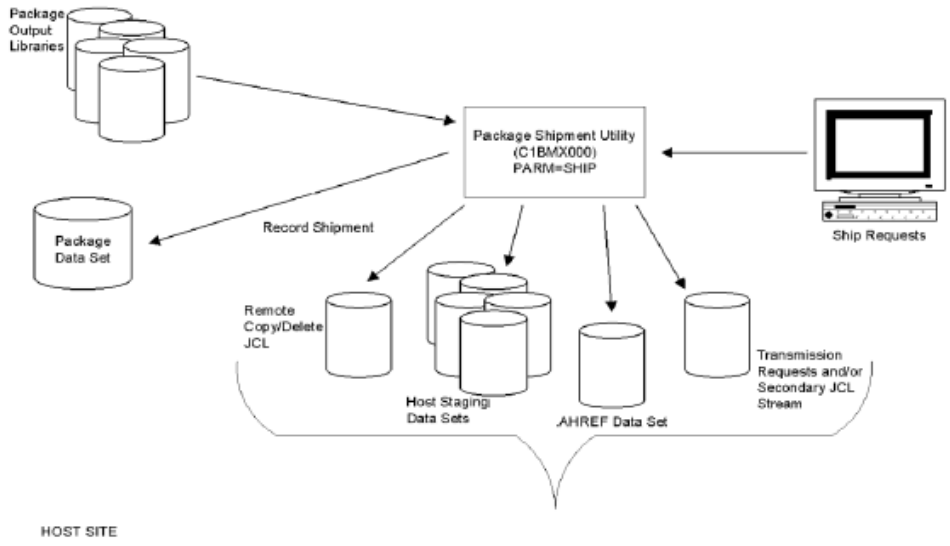
```
//*********************************************************************
//* CREATE SEQUENTIAL FILE FROM PDS MEMBERS
//*********************************************************************
//SCR020   EXEC PGM=IEBPTPCH
//SYSPRINT DD   SYSOUT=*
//SYSUT1   DD   DISP=SHR,    REMOTE STAGING DATA SET FOR SCRIPT
//         DSN=&ZZNATCMD
//SYSUT2   DD   DSN=PUBLIC.MORMI08.NATCMDT,
//         DISP=(,CATLG),
//         SPACE=(CYL,(1,1),RLSE),
//         UNIT=SYSDA,
//         DCB=(RECFM=FBA,LRECL=81)
//SYSIN    DD  *
 PUNCH TYPORG=PO
//*
//*********************************************************************
//* REMOVE LEADING CONTROL CHARACTER AND SEPARATOR LINES
//*********************************************************************
//SCR030   EXEC PGM=SORT
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//REPORT1  DD   SYSOUT=*
//SORTIN   DD   DISP=SHR,DSN=PUBLIC.MORMI08.NATCMDT
//SORTOUT  DD   DSN=PUBLIC.MORMI08.NATCMDS,
//         DISP=(,CATLG),
//         UNIT=SYSDA,
//         SPACE=(CYL,(1,1)),
//         DCB=(RECFM=FB,LRECL=80)
//SORTWK01 DD  UNIT=DISK,SPACE=(CYL,(5,5))
//SORTWK02 DD  UNIT=DISK,SPACE=(CYL,(5,5))
//SORTWK03 DD  UNIT=DISK,SPACE=(CYL,(5,5))
//SYSIN    DD  *
  OMIT COND=((2,13,CH,EQ,C'MEMBER NAME  '))
  SORT FIELDS=COPY
  INREC FIELDS=(1:2,80)
//*
//GENER1   EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
//SYSUT1   DD   DATA,DLM='$$'
&RJOBCARDS
$$
//         DD   DISP=SHR,DSN=PUBLIC.MORMI08.NATCMDS
//SYSUT2   DD   SYSOUT=(A,INTRDR)
@ENDIF
```

Host Package Shipment Job Steps

When you submit the shipment queue, CA Endevor SCM builds and submits a JCL stream to ship the packages. The remainder of this section describes the steps in this shipment JCL.

How to Build and Stage the Shipment

As shown in the following, CA Endevor SCM uses the C1BMXJOB and C1BMXLIB skeletal JCL to build this step.



As previously illustrated, when it executes this step, the ship utility builds a shipment consisting of the following:

- A staging data set for each library involved in the package execution.
- JCL for execution at the remote site, consisting of an IEBCOPY step, an IDCAMS (delete) step, and confirmation steps.
- Data transmission utility commands and/or JCL to transmit the data sets and execute the remote copy/delete job. (Not applicable for local transmissions.)
- A file of correspondences between host and remote production and staging data sets.
- Complementary files and JCL, if necessary.

The ship utility then populates the staging data sets, records the shipment, and passes the entire shipment to Step 2.

Staging Data Sets

A staging data set prefix is defined for each destination. This allows the staging data sets to be catalogued. The ship utility generates the remainder of the data set name. The format is as follows:

`prefix.`*`Dyymmdd.Thhmmss.destination.suffix`*

prefix

The user-defined prefix for the host or remote site on the Create/Modify Definition panel, or the default value TSOuserid.NDVR.

Dyymmdd

The date the shipment queue was submitted.

Thhmmss

The time the shipment queue was submitted.

destination

The destination name.

suffix

The kind of shipment member. Possible suffixes are:

AxJOB—Identifies a host (AHJOB) or remote (ARJOB) copy/delete/confirm job stream.

CxJOB—Identifies a host (CHJOB) or remote (CRJOB) copy/delete/confirm job stream for complementary data sets

.AxREF—Identifies a host (AHREF) or remote (ARREF) data set name and or USS path cross-reference file.

AxPAX—Identifies the a host (AHPAX) or remote (ARPAX) Archive files that are used to contain USS files and their complementaries, if any, in the shipment. This file is optional and is only present if the shipment request includes USS objects.

AxUCD—Identifies the host (AHUCD) or remote (ARUCD) commands stream that will be perform the un-pax, and copy commands to move shipped USS objects to the final location. This file is optional and is only present if the shipment request includes USS objects.

.AHnnn—Identifies a host shipment data set, where nnn is a sequential number starting at 001.

.CHnnn—Identifies a host complementary data set, where nnn is a sequential number starting at 001.

.ARnnn—Identifies a remote shipment data set, where nnn is a sequential number and corresponds to its host counterpart AHnnn.

.CRnnn—Identifies a remote complementary data set, where nnn is a sequential number and corresponds to its host counterpart CHnnn.

.SHnnn—Identifies a host shipment script data set, where nnn is a sequential number starting at 001.

.THnnn—Identifies a host complementary script data set, where nnn is a sequential number starting at 001.

.SRnnn—Identifies a remote shipment script data set, where nnn is a sequential number and corresponds to its host counterpart SHnnn.

.TRnnn—Identifies a remote complementary script data set, where nnn is a sequential number and corresponds to its host counterpart THnnn.

Examples of staging data set names include the following:

- *userid*.NDVR.D071130.T142532.BOSTON.AH0000034
- *userid*.NDVR.D071130.T143515.CHICAGO.CRJOB
- *userid*.NDVR.D071130.T145216.BOSTON.ARREF

Remote Execution JCL

The data set name of the JCL for remote execution has the suffix .AHJOB. The JCL consists of up to four job steps, described as follows:

1. Copying package outputs from remote staging data sets to production data sets, using IEBCOPY.
2. Deleting members from the production data sets that were also deleted on the host.
3. Confirming the copy/delete procedure.
4. Optional. Deleting the staging data sets.

   If the ship utility also builds complementary data sets, it also builds JCL for those data sets giving them a data set name suffix of .CHJOB.

Remote JCL Execution Commands

You can execute the remote JCL automatically or at the discretion of the remote site (manually). The choice is made by tailoring the "E" model transmission control members (#PSXCOME, #PSBDT1E, #PSLOCLE, #PSNFTPE. #PSNWDME, or #PSBAT2E). Complementary data set JCL (.CHJOB) can only be executed manually.

Note: For more information, see Creating Model Transmission Control Statements (see page 110).

Data Set Correspondences File

The data set name for the following data set correspondences has the suffix .AHREF. For every data set involved in the package execution, this file specifies both production names and staging names at the host and remote sites.

An example follows:

```
 SHIPMENT DATASETS
  HOST LIBRARY:    BST.XDVRC1S1.LISTINGS
  HOST STAGING:    DA1ME10.D10322.T164235.BOSTON1.AH003
 REMOTE STAGING:    DA1ME10.D10322.T164235.BOSTON1.AR003
 REMOTE LIBRARY:    BST.XDVRC1S1.LISTINGS
*
```

```
 SHIPMENT DATASETS
  HOST LIBRARY:   BST.XDVRC1S1.LKEDLIB
  HOST STAGING:   DA1ME10.D10322.T164235.BOSTON1.AH004
 REMOTE STAGING:   DA1ME10.D10322.T164235.BOSTON1.AR004
 REMOTE LIBRARY:   BST.XDVRC1S1.LKEDLIB
*


 SHIPMENT DATASETS
  HOST LIBRARY:   BST.XDVRC1S1.LKEDLIST
  HOST STAGING:   DA1ME10.D10322.T164235.BOSTON1.AH006
 REMOTE STAGING:   DA1ME10.D10322.T164235.BOSTON1.AR006
 REMOTE LIBRARY:   BST.XDVRC1S1.LKEDLIST
*


 SHIPMENT DATASETS
  HOST LIBRARY:   BST.XDVRC1S1.LOADLIB
  HOST STAGING:   DA1ME10.D10322.T164235.BOSTON1.AH005
 REMOTE STAGING:   DA1ME10.D10322.T164235.BOSTON1.AR005
 REMOTE LIBRARY:   BST.XDVRC1S1.LOADLIB
*


 SHIPMENT DATASETS
  HOST LIBRARY:   BST.XDVRC1S1.OBJLIB
  HOST STAGING:   DA1ME10.D10322.T164235.BOSTON1.AH002
 REMOTE STAGING:   DA1ME10.D10322.T164235.BOSTON1.AR002
 REMOTE LIBRARY:   BST.XDVRC1S1.OBJLIB
```

Shipments Contents

By the end of Step 1, the ship utility has built a shipment consisting of the following:

- Package outputs (.AHnnn)
- Remote JCL (.AHJOB)
- Data set cross-reference file (.AHREF)

If the shipment contained USS files, the following are *also* included:

- An Archive file (.AHPAX)
- BPXBATCH commands to perform the unarchive and copy functions.

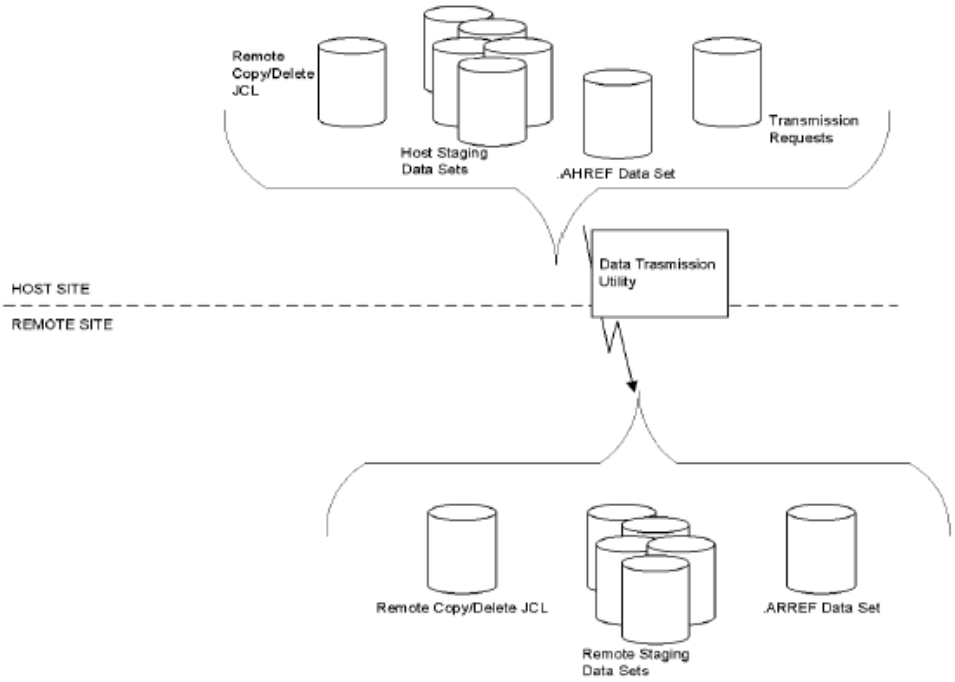
If the shipment contained script files, the following are also included:

- Script files (.SHnnn)

If complementary data sets have been requested, the shipment also contains the remote JCL for the complementary files (.CHJOB) and if appropriate, the complementary script files).

How to Transmit the Shipment

As shown in the following, CA Endevor SCM uses one of the skeletal JCL members C1BMXCOM, C1BMXBD1, C1BMXBDT, C1BMXLOC, C1BMXFTP, or C1BMXNDM to build this job step for ISPF requested shipments or one of the following JCL/Proclib members for API requested shipments; SHIPBDT1, SHIPBDT2, SHIPCONN, SHIPLOCL, SHIPNVFT,or SHIPXCOM.



As previously illustrated, XCOM (C1BMXCOM), BDT Version 2 (C1BMXBDT), and CONNECT:Direct (C1BMXNDM) execute in this step. The transmission program performs the following steps:

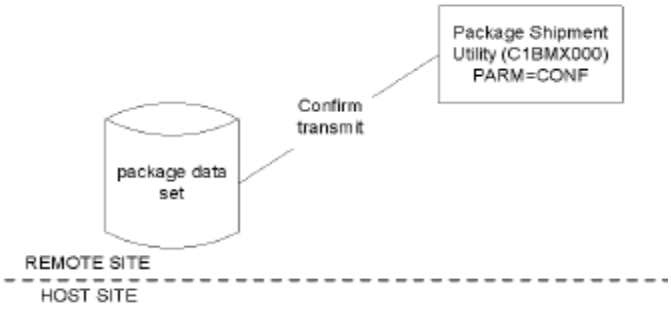
1. Reads its control cards (generated in Step 1).
2. If there are any script files, they are staged and any "Destination Symbols" are resolved.
3. Transmits the host staging data sets to the remote sites where it builds remote staging data sets.
4. Executes the remote copy/delete JCL (.AxJOB) if the skeletal JCL (BDT and CONNECT:Direct) or model transmission control member (XCOM) are set up for automatic execution.

BDT via NJE/NJI (C1BMXBD1) and NetView FTP (C1BMXFTP) are executed in a secondary job stream built in Step 1. This step consists of writing that job stream to an internal reader. The first step of the secondary job stream performs the same functions as listed previously.

Local transfers (C1BMXLOC) do not execute a physical transmission. The copy/delete JCL is submitted for execution if the model transmission control member is set up for automatic execution.

How to Confirm the Transmission

As shown in the following, CA Endevor SCM uses the skeletal JCL members C1BMXHCN and C1BMXLIB to build this job step for ISPF transmissions, or JCL/Proclib member SHIPHCN for API shipments..

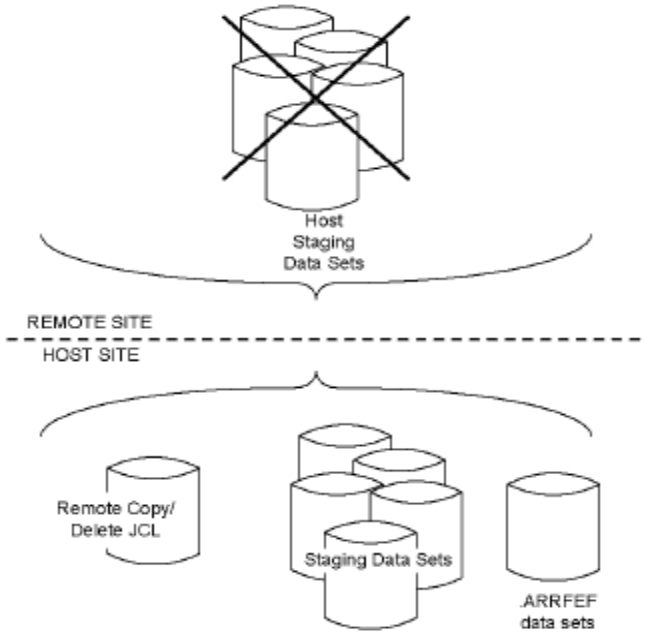


As previously illustrated, during this step, the ship utility records confirmation of the transmission to the remote site.

Note: There is an after-confirm exit that can be used, in conjunction with the notification utility, to notify users of the completion of the shipment.

How to Delete the Staging Data Sets

As shown in the following, CA Endevor SCM uses the C1BMXEND skeletal JCL to build this job step.

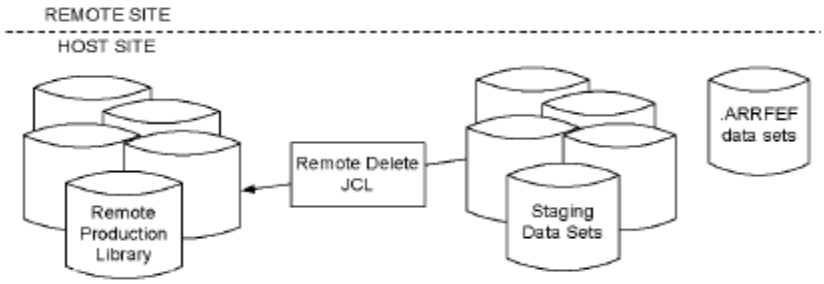


As previously illustrated, if the DISPOSITION field for the host staging data set names has the value DELETE, the ship utility deletes the host staging data sets.

How the Remote Copy/Delete Job Steps Work

Remote JCL is transmitted in a data set with the suffix .AHJOB. If the shipment includes complementary data sets, the remote JCL for these data sets has the suffix .CHJOB. Remote JCL consists of the following job steps:

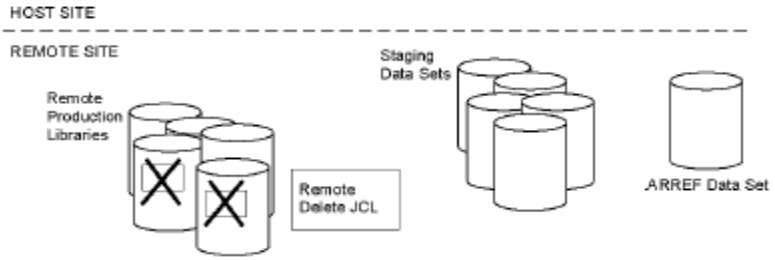
1. Execution of the "Before" ship script file steps from #RJNDVRB (if any).
2. Copying package outputs from staging data sets to production data sets, using IEBCOPY.



3. If the package contained any USS objects, then an additional step (BPXBATCH) invokes the command stream (ARUCD) to unarchive the USS objects to the staging directories, and copies each file to the final location.

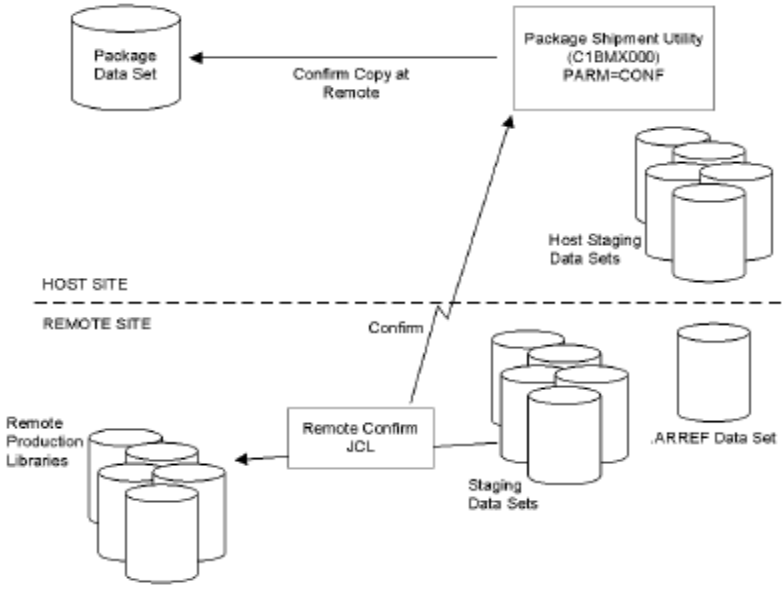
   Note: The actual syntax built performs a delete (rm) for each output before performing a copy (cp) for each file. This ensures that date and timestamps on the remote files match the corresponding generate (move) times on the host system.

4. Deleting members from the production data sets or paths that were also deleted on the host during package execution or backout.

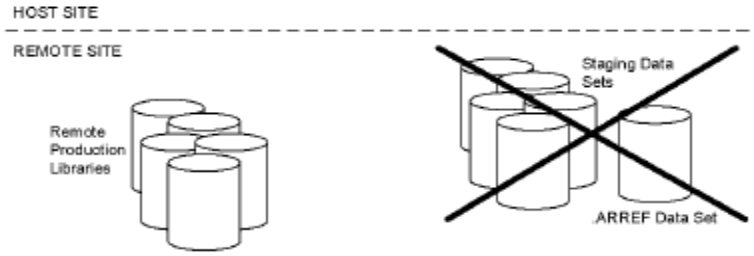


5. Execution of the "After" ship script file steps from #RJNDVRA (if any).

6. Confirming the copy/delete procedure.



7. (Optional) Deleting the staging and cross reference data sets. This step is included only if the disposition of the staging data sets on the destination record used for the shipment is **delete**.



Example: Use Remote JCL

A sample of remote JCL follows:

```
//JOBNAME JOB (ACCOUNT),'NAME'
//*
//* *-----------------------* ISPSLIB(C1BMXCOP)
//* * REMOTE SITE JOBSTEP TO COPY MEMBERS WHICH WERE
//* * MODIFIED BY THE EXECUTION OF THE PACKAGE
//* *--------------------------------*
//COPY  EXEC PGM=IEBCOPY
//SYSUT3  DD UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT4  DD UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSPRINT DD SYSOUT=*


//IAR001  DD DISP=SHR,DSN=TSOUSER.NDVR.D11119.T123747.DESTNDM.AR001
//OAR001  DD DISP=OLD,DSN=TEST.IMRENV1.SRCOUT1
//IAR002  DD DISP=SHR,DSN=TSOUSER.NDVR.D11119.T123747.DESTNDM.AR002
//OAR002  DD DISP=OLD,DSN=TEST.IMRENV1.OBJLIB1
//IAR003  DD DISP=SHR,DSN=TSOUSER.NDVR.D11119.T123747.DESTNDM.AR003
//OAR003  DD DISP=OLD,DSN=TEST.UTILS1.LISTINGS
//SYSIN  DD *
 COPY  OUTDD=OAR002,INDD=((IAR002,R))
 COPY  OUTDD=OAR001,INDD=((IAR001,R))
 COPY  OUTDD=OAR003,INDD=((IAR003,R))
//* *-----------------------* ISPSLIB(C1BMXDEL)
//* * REMOTE SITE JOBSTEP TO DELETE MEMBERS WHICH
//* * WERE DELETED BY THE EXECUTION OF THE PACKAGE
//* *--------------------------------*
```

```
//DELETE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN  DD *
 DELETE 'TSOUSER.NDVR.D11119.T123747.DESTNDM.AR002' NONVSAM
 DELETE 'TSOUSER.NDVR.D11119.T123747.DESTNDM.AR001' NONVSAM
 DELETE 'TSOUSER.NDVR.D11119.T123747.DESTNDM.AR003' NONVSAM
 DELETE 'TSOUSER.NDVR.D11119.T123747.DESTNDM.ARJOB' NONVSAM
 DELETE 'TSOUSER.NDVR.D11119.T123747.DESTNDM.ARREF' NONVSAM
//CONFCOPY EXEC PGM=IEBGENER      EXECUTED AT THE REMOTE SITE
//SYSUT1  DD DATA,DLM=$$       JOB SHIPPED BACK TO HOST


//TSOUSERP JOB (1111),'WESTBORO',CLASS=A,
// MSGCLASS=X,NOTIFY=TSOUSER
/*ROUTE PRINT U101
/*ROUTE XEQ HOSTNODE
//CONFCOPY EXEC PGM=NDVRC1,
//  PARM='C1BMX000,19911119,12374712,CONF,RCPY,EX,****,DESTNDM '
//*
//* *----------------------* ISPSLIB(C1BMXLIB) *
//*


//* *============================================================*
//* * STEPLIB, CONLIB, MESSAGE LOG AND ABEND
DATASETS
//* *============================================================*
//*
//STEPLIB DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTH
//CONLIB  DD DSN=TEST.NDVR36B.ZAPLOAD,DISP=SHR
//     DD DSN=TEST.C19109.CONLIB,DISP=SHR
//*
//SYSABEND DD SYSOUT=*
//C1BMXLOG DD SYSOUT=*   *** MESSAGES, ERRORS, RETURN CODES ********
//* *----------------------* C1BMXRCN (CONT.) *


$$
//SYSUT2  DD SYSOUT=(A,INTRDR)
//SYSPRINT DD SYSOUT=*
//SYSIN  DD DUMMY
//*
```

```
//CONFABND EXEC PGM=IEBGENER,COND=ONLY EXECUTED AT THE REMOTE SITE
//SYSUT1  DD DATA,DLM=$$       JOB SHIPPED BACK TO HOST
//TSOUSERP JOB (1111),'WESTBORO',CLASS=A,
// MSGCLASS=X,NOTIFY=TSOUSER
/*ROUTE PRINT U101
/*ROUTE XEQ HOSTNODE
//CONFCOPY EXEC PGM=NDVRC1,
//  PARM='C1BMX000,19911119,12374712,CONF,RCPY,AB,****,DESTNDM '
//*
//* *---------------------* ISPSLIB(C1BMXLIB) *
//*


//* *=============================================================*
//* * STEPLIB, CONLIB, MESSAGE LOG AND ABEND DATASETS
//* *=============================================================*
//*
//STEPLIB DD DISP=SHR,DSN=iprfx.iqual.CSIQAUTH
//CONLIB  DD DSN=TEST.NDVR36B.ZAPLOAD,DISP=SHR
//     DD DSN=TEST.C19109.CONLIB,DISP=SHR
//*
//SYSABEND DD SYSOUT=*
//C1BMXLOG DD SYSOUT=*   *** MESSAGES, ERRORS, RETURN CODES ********


//* *---------------------* C1BMXRCN (CONT.) *
$$
//SYSUT2  DD SYSOUT=(A,INTRDR)
//SYSPRINT DD SYSOUT=*
//SYSIN  DD DUMMY
//* **** END OF JCL STREAM ****
```

How to Build, Track and Confirm Shipments

Administering Package Ship involves building, tracking, and confirming shipments. The first step in shipping a package is to build a shipment request. You can build ship requests either in foreground or in batch. This section describes how to build shipments in foreground.

Note: For more information about how to code package shipment requests in batch, see Package Shipment SCL.

When you press ENTER after building a shipment, CA Endevor SCM places that shipment in a request queue. You can do the following:

- Display the queue, reset it (deleting all shipment requests), or submit it.
- Ship the packages in the queue by submitting the queue.

You build ship requests in foreground from the Package Shipment panel. To access the Package Shipment panel, type **6** (SHIP) in the OPTION field on the Package Options Menu; press ENTER. CA Endevor SCM displays the Package Shipment panel. On the Package Shipment panel enter the appropriate information and type **1** (BUILD SHIPMENT REQUEST); press ENTER. CA Endevor SCM displays a Package Selection List and/or a Destination Selection List, then a Confirm Shipment panel. You place a shipment in the shipment queue from the Confirm Shipment panel.

The procedures that follow explain how to build requests to ship the following:

- One package to one destination
- One package to multiple destinations
- Multiple packages to one destination

Ship One Package to One Destination

To ship one package to one destination

1. Type **1** in the OPTION field on the Package Shipment panel, along with all required and any optional information you desire. Press ENTER.
2. If the Confirm Shipment panel displays, proceed to Step 3. Otherwise select the package and/or destination from the selection lists that appear.

   When you press ENTER on the last selection list, the Confirm Shipment panel displays.
3. Review the information on the Confirm Shipment panel and:
   - Press END key to return to the previous panel, or
   - Type **SH** and press ENTER to place the ship request in the request queue.

   Note: For more information, see Confirm a Package for Shipment (see page 165).
4. Press END key until you return to the panel you need to perform your next action.

Ship One Package to Multiple Destinations

To ship one package to multiple destinations

1. Type **1** in the OPTION field on the Package Shipment panel, as well as all required and any optional information you want and press ENTER.
2. If the Confirm Shipment panel displays, proceed to Step 3. Otherwise select the package and/or destinations from the selection lists that appear. When you press ENTER on the last selection list, the Confirm Shipment panel displays.
3. Review the information on the Confirm Shipment panel, then type **SH** and press ENTER to place the ship request in the request queue. When you press ENTER, the DESTINATION field changes to display the next destination you have selected.

   Note: For more information, see Confirm a Package for Shipment (see page 165).
4. Repeat Step 3 until the Destination Selection List appears when you press ENTER. This indicates that you have placed shipments for all selected destinations in the shipment queue.
5. You can now:
   - Select another destination, then press ENTER to display the Confirm Shipment panel. Proceed from Step 3.
   - Press END key until you return to the panel you need to perform your next action.

Ship Multiple Packages to One Destination

To ship multiple packages to one destination

1. Type **1** in the OPTION field on the Package Shipment panel, as well as all required and any optional information you desire and press ENTER.
2. If the Confirm Shipment panel displays, proceed to Step 3. Otherwise select the packages and/or destination from the selection lists that appear.

   When you press ENTER on the last selection list, the Confirm Shipment panel displays.
3. Review the information on the Confirm Shipment panel, then type **SH** and press ENTER to place the ship request in the request queue and return to the Destination Selection List. The name of the first selected package displays in the WHICH field on this panel.

   Note: For more information, see Confirm a Package for Shipment (see page 165).
4. Press END key on the Destination Selection List to display the second selected package. Then select a destination for this package and press ENTER to display the Confirm Shipment panel. Repeat Step 3.
5. Repeat Steps 3 and 4 until either the Package Selection List or Package Shipment panel appears when you press END key. This indicates that you have placed shipments for all selected packages in the shipment queue.
6. You can now:
   - Continue building shipments.
   - Press END key until you return to the panel you need to perform your next action.

The Package Shipment Panel

You can provide the following information when building a package ship request:

Package ID

Required. The 1- to 16-character ID of the package you want to ship.

Destination

Required. The 1- to 7-character name of the destination which is to receive the package (or backouts).

Pkg/Backout

Default is **P**, package. Indicates whether you want to ship package outputs or backout members. Acceptable values are:

P-Ship package outputs.

B-Ship backout members.

Status Date Range

Optional. Allows you to specify a range of dates to be covered by the shipment status list. Date format is mmddyy.

Status Sort Order (Required)

You can sort the shipment status list by shipment date, by destination ID, and/or by package ID. You must specify the order in which these sorts are performed. Acceptable values are:

1-Perform this sort first.

2-Perform this sort second.

3-Perform this sort last.

The Package Selection List

This panel allows you to select a package for shipment (**S**) and/or to display a package before selecting it (**D**). For each package, the panel displays the following information:

Package

The up to 16 character package ID

Status

The status of the package. The status must be executed in order to ship the package.

Description

The up to 40 character package description.

The Destination Selection List

This panel allows you to select a destination (**S**) for a package. The package to be shipped displays in the S - SELECT DESTINATION TO RECEIVE: field. For each destination, the panel displays the following information:

DEST-ID

Identifies the destination name.

MESSAGE

Identifies a literal or message ID indicating an action was performed against this destination (for example, *MODIFIED, *DELETED, and so on).

TRAN-METHOD

Identifies the data transmission utility used for this destination.

RMT-NODENAME

Identifies the data transmission utility nodename of the destination.

DESCRIPTION

Describes the destination.

REMOTE-IPNAME

Identifies the site to which package outputs are to be shipped.

IPPORT

Identifies the XCOM SERVPORT specification of the target server.

Note: If a remote IPNAME is specified for a destination, you can scroll to the right to view the REMOTE IPNAME and IPPORT. ESORT is supported for all column headings.

Press END key from this panel to perform the following actions:

- Display the next selected package in the S - SELECT DESTINATION TO RECEIVE: field, if multiple packages were selected and not all have been placed in the shipment queue.
- Return to the Package Selection List or Package Shipment panel if there are no more selected packages.

Confirm a Package for Shipment

To place a package in a shipment queue, you must confirm the shipment on the Confirm Shipment panel, which is displayed for each package and destination pair. This panel lets you to review and change your choice of outputs (package or backout) and the shipment command file prefix for the XCOM or CONNECT:DIRECT transmission methods, before placing the package and destination pair in the shipment queue.

To confirm a package for shipment

1. Review the shipment information on the Confirm Shipment panel and change the values in the following fields, if necessary.

   Note: If you do not want to place the package in the shipment queue at this time, press the End key to return to the immediately prior panel.

   - **PKG/BACKOUT**—This field allows you to change shipment contents before placing the shipment in the queue. Valid values are:
     - **PACKAGE**—Shipment contains package outputs.
     - **BACKOUT**—Shipment contains package backout members.
   - **SHIPMENT COMMAND FILE PREFIX**—This field specifies the 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 field is ignored. If a MODHLI parameter value is coded in the C1DEFLTS table, that value appears in this field the first time this panel is displayed within the session. If the MODHLI parameter value is blank, the TSO user ID value appears in this field. If you clear the field, then the MODHLI or the TSO user ID, in that order, will be used as the data set name prefix.

     The value used will appear in this field the next time the panel is displayed until you leave the Package Options. The next time the Package Options are selected, the initial defaults will be set.

2. Type **SH** in the OPTION field, then press Enter.

   The shipment displayed on the panel is placed in the shipment queue one of the following occurs:

   - The next selected destination appears in the DESTINATION field, if you selected multiple destinations for this package and not all package and destination pairs have been placed in the shipment queue.
   - The Destination Selection panel appears, if you selected multiple packages and not all packages have been placed in the shipment queue.

Shipment Tracking and Confirmation

A record of shipment is created when the shipment staging utility has finished creating all the host staging data sets. At this time the package and destination are recorded along with the shipment staging return code. These can be viewed by using option **5,** DISPLAY SHIPMENT STATUS, of the Package Shipment panel.

Two other points in the process of shipping are tracked: the execution of the data transmission utility and the execution of the remote COPY/DELETE/CONFIRM job. These points are tracked by executing the shipment confirmation utility as a conditional job step after the execution of the data transmission utility (at the host site) and after the execution of the IEBCOPY/IDCAMS job (at the remote site).

The Confirmation Steps

There are two ISPF Skeletal JCL members which deal with shipment confirmation for ISPF initiated shipments. C1BMXHCN handles confirmation of the data transmission job step which executes at the host site. C1BMXRCN handles confirmation of the IEBCOPY/IDCAMS job which executes at the remote site. There is also a corresponding pair of JCL/PROCLIB members which are called to perform confirmations for API shipments (SHIPHCN and SHIPRCN). Left unmodified, these steps would leave a generic record of the data transmission or COPY/DELETE. The record of shipment would be marked either EXEC'D or ABEND under the "HOST TRANS" or "REMOTE MOVE" columns of the Package Shipment Status panel.

If a more precise record of these events is required, the number of conditional confirmation steps can be increased, status of script file exececution steps can be returned, and the shipment confirmation utility parameter can be changed to report different things.

The confirmation parameter is made up of a symbolic prefix (resolved by ISPF facilities), a root, and a symbolic suffix (resolved by the shipment staging utility)

```
//STEPNAME EXEC PGM=NDVRC1,PARM='&VNBCPARM.,CONF,pppp,op,code,********',
//         COND=(see_example)
```

Where:

- Pppp-Is the point of confirmation. This is a four-character string which can have the following values
  - HXMT-Host transmission confirmation.
  - RCPY-Remote COPY/DELETE confirmation.
  - SCRP- Remote Script confirmation
- Op-Is the algebraic operator associated with the return code of the step being confirmed. The valid operators are:
  - EQ-Equal.
  - GE-Greater than or equal.
  - LE-Less than or equal.
  - GT-Greater than.
  - LT-Less than.

  There are also two generic non-algebraic operators:
  - EX-The job executed (return code not available).
  - AB-The job abended (abend code not available).
- Code-Is the code to be reported. Usually this would be a four-digit number which bears a relationship to the step being confirmed.

Example: Modify C1BMXHCN to Record Specific Codes

Here is a sample of how C1BMXHCN could be modified to report the following return codes: RC=00, RC=04, RC=08, RC>=12, ABEND.

```
//CONFGE12 EXEC PGM=NDVRC1,REGION=4096K,COND=(12,GT,&VNBXSTP),
//  PARM='&VNBCPARM,CONF,HXMT,GE,0012,********'
//C1BMXDTM DD DSN=&&&&XDTM,DISP=(MOD,PASS),SPACE=(TRK,(1,0)),
//      DCB=(RECFM=F,LRECL=80,BLKSIZE=3120,DSORG=PS),
//      UNIT=tdisk
//*
)IM C1BMXLIB OPT
//*
//*
//CONFEQ08 EXEC PGM=NDVRC1,REGION=4096K,COND=(08,NE,&VNBXSTP),
//  PARM='&VNBCPARM,CONF,HXMT,EQ,0008,********'
//C1BMXDTM DD DSN=&&&&XDTM,DISP=(MOD,PASS),SPACE=(TRK,(1,0)),
//      DCB=(RECFM=F,LRECL=80,BLKSIZE=3120,DSORG=PS),
//      UNIT=tdisk
//*
)IM C1BMXLIB OPT
//*
//*
//CONFEQ04 EXEC PGM=NDVRC1,REGION=4096K,COND=(04,NE,&VNBXSTP),
//  PARM='&VNBCPARM,CONF,HXMT,EQ,0004,********'
//C1BMXDTM DD DSN=&&&&XDTM,DISP=(MOD,PASS),SPACE=(TRK,(1,0)),
//      DCB=(RECFM=F,LRECL=80,BLKSIZE=3120,DSORG=PS),
//      UNIT=tdisk
//*
)IM C1BMXLIB OPT
//*
//*
//CONFEQ00 EXEC PGM=NDVRC1,REGION=4096K,COND=(00,NE,&VNBXSTP),
//  PARM='&VNBCPARM,CONF,HXMT,EQ,0000,********'
//C1BMXDTM DD DSN=&&&&XDTM,DISP=(MOD,PASS),SPACE=(TRK,(1,0)),
//      DCB=(RECFM=F,LRECL=80,BLKSIZE=3120,DSORG=PS),
//      UNIT=tdisk
//*
)IM C1BMXLIB OPT
//*
//*
//CONFABND EXEC PGM=NDVRC1,REGION=4096K,COND=ONLY,
//  PARM='&VNBCPARM,CONF,HXMT,AB,****,********'
//C1BMXDTM DD DSN=&&&&XDTM,DISP=(MOD,PASS),SPACE=(TRK,(1,0)),
//      DCB=(RECFM=F,LRECL=80,BLKSIZE=3120,DSORG=PS),
//      UNIT=tdisk
//*
)IM C1BMXLIB OPT
//*
```

Example: Modify C1BMXRCN to Record Specific Codes

Here is a sample of how C1BMXRCN could be modified to report the following return codes: RC=00, RC=04, RC=08, RC>=12, ABEND. Note that the COND= parameter is on the IEBGENER execute card, not on the NDVRC1 execute card. $DEST_ID is resolved by the shipment staging utility.

```
//CONFGT12 EXEC PGM=IEBGENER,COND=(12,GT)
//SYSUT1  DD DATA,DLM=$$       JOB SHIPPED BACK TO HOST
)IM C1BMXHJC
//CONFCOPY EXEC PGM=NDVRC1,
//       PARM='&VNBCPARM,CONF,RCPY,GE,0012,$DEST_ID'
)IM C1BMXLIB
$$
//SYSUT2  DD SYSOUT=(INTRDR,A)
//SYSPRINT DD SYSOUT=*
//SYSIN  DD DUMMY
//*
//* *-------------------------------*
//*
//CONFEQ08 EXEC PGM=IEBGENER,COND=(08,NE)
//SYSUT1  DD DATA,DLM=$$       JOB SHIPPED BACK TO HOST
)IM C1BMXHJC
//CONFCOPY EXEC PGM=NDVRC1,
//       PARM='&VNBCPARM,CONF,RCPY,EQ,0008,$DEST_ID'
)IM C1BMXLIB
$$
//SYSUT2  DD SYSOUT=(INTRDR,A)
//SYSPRINT DD SYSOUT=*
//SYSIN  DD DUMMY
//*
//* *-------------------------------*
//*
```

```
//CONFEQ04 EXEC PGM=IEBGENER,COND=(04,NE)
//SYSUT1  DD DATA,DLM=$$       JOB SHIPPED BACK TO HOST
)IM C1BMXHJC
//CONFCOPY EXEC PGM=NDVRC1,
//       PARM='&VNBCPARM,CONF,RCPY,EQ,0004,$DEST_ID'
)IM C1BMXLIB
$$
//SYSUT2  DD SYSOUT=(INTRDR,A)
//SYSPRINT DD SYSOUT=*
//SYSIN  DD DUMMY
//*
//* *-------------------------------*
//*
//CONFEQ00 EXEC PGM=IEBGENER,COND=(00,NE)
//SYSUT1  DD DATA,DLM=$$       JOB SHIPPED BACK TO HOST
)IM C1BMXHJC
//CONFCOPY EXEC PGM=NDVRC1,
//       PARM='&VNBCPARM,CONF,RCPY,EQ,0000,$DEST_ID'
)IM C1BMXLIB
$$
//SYSUT2  DD SYSOUT=(INTRDR,A)
//SYSPRINT DD SYSOUT=*
//SYSIN  DD DUMMY
//*
//* *-------------------------------*
//*
//CONFABND EXEC PGM=IEBGENER,COND=ONLY EXECUTED AT THE REMOTE SITE
//SYSUT1  DD DATA,DLM=$$       JOB SHIPPED BACK TO HOST
)IM C1BMXHJC
//CONFCOPY EXEC PGM=NDVRC1,
//       PARM='&VNBCPARM,CONF,RCPY,AB,****,$DEST_ID'
)IM C1BMXLIB
$$
//SYSUT2  DD SYSOUT=(INTRDR,A)
//SYSPRINT DD SYSOUT=*
//SYSIN  DD DUMMY
```

Note: If you perform shipments using both the ISPF interface and the API interface be sure to apply your customizations to both the models in the CSIQOPTN library and the JCL/PROCLIB supplied in the CSIQJCL library.

The Request Queue

CA Endevor SCM stores ship requests in a queue until they are executed. The queue is active only for the current CA Endevor SCM session.

CA Endevor SCM displays the queue on the Queued Shipment List.

You can perform the following actions:

- Display the Queued Shipment List by typing **2** in the OPTION field on the Package Shipment panel and pressing ENTER.
- Reset the Queued Shipment List by typing **4** in the OPTION field on the Package Shipment panel and pressing ENTER. This deletes all shipments from the queue.
- Submit the request queue by typing **3** in the OPTION field on the Package Shipment panel and pressing ENTER. When you submit a shipment request, CA Endevor SCM automatically resets the queue.

Display Shipment Status

After you have submitted the request queue, CA Endevor SCM displays the status of the shipments on the Package Shipment Status panel. You access the Package Shipment Status panel by typing **5** in the OPTION field on the Package Shipment panel, then pressing ENTER.

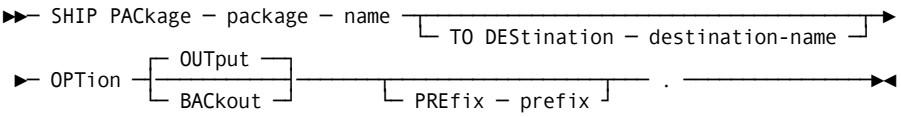
Use the scroll left/right (PF10/PF11) to alternate between the available views; showing either the local (host) shipment status or the remote copy, and script execution status.

Note: The shipment records displayed are filtered according to the Package and Destination values supplied on the Shipment panel. To change the filter, hit PF3 to back out a level and enter a value or mask, or leave these fields blank to display all shipment records.

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.

Package Ship SCL Examples

When executing the Package Ship, only a single option can be specified for a given package going to a given destination. That is, the following request would be treated as an error:

SHIP PACKAGE ABCD TO DESTINATION CHICAGO OPTION OUTPUT.

SHIP PACKAGE ABCD TO DESTINATION CHICAGO OPTION BACKOUT.

You can specify different options for different package/destination combinations, however. The next request is valid:

SHIP PACKAGE ABCD TO DESTINATION CHICAGO OPTION OUTPUT.

SHIP PACKAGE ABCD TO DESTINATION BOSTON OPTION BACKOUT.

On occasion, a module can be affected by the execution of two different packages. If you try to ship output members from one package and backout members from the other package to the same destination, you receive an error message. For example, assume a module has been affected by the execution of PKG01 and PKG02. A request similar to that shown next is invalid:

SHIP PACKAGE PKG01 TO DESTINATION BOSTON OPTION OUTPUT.

SHIP PACKAGE PKG02 TO DESTINATION BOSTON OPTION BACKOUT.

You can ship the two packages to the same destination if you use the same option, however. Therefore, either of the next requests would be acceptable:

SHIP PACKAGE PKG01 TO DESTINATION BOSTON OPTION OUTPUT.

SHIP PACKAGE PKG02 TO DESTINATION BOSTON OPTION OUTPUT.

or

SHIP PACKAGE PKG01 TO DESTINATION BOSTON OPTION BACKOUT.

SHIP PACKAGE PKG02 TO DESTINATION BOSTON OPTION BACKOUT.

Note: The sequence of the Package Shipment Utility requests is unimportant. The utility looks at the package execution dates.

SET Statements for Package Ship SCL

SET statements are global default statements, and establish values for subsequent SHIP statements. If a parameter is required (or used) but not coded, CA Endevor SCM looks for that information in a previous SET statement.

Note: If you code a SET statement but enter similar information in the SHIP clause, the value in the SHIP clause will override the SET statement value.

You can use two SET statements with the Package Shipment Utility:

SET DESTINATION *destination-name*

The SET DESTINATION clause applies to each subsequent SHIP statement that does not contain a destination. Because destination is required, you must have a SET statement (previously) coded if you omit the TO DESTINATION clause.

The destination name specified here will be used until the system encounters another SET DESTINATION statement or a CLEAR DESTINATION statement, or processing ends.

SET OPTION OUTPUT/BACKOUT

The SET OPTION clause applies to each subsequent SHIP statement that does not specify an option. The option you indicate in the SET clause is used until the system encounters another SET OPTION statement or a CLEAR OPTIONS statement, or processing ends.

Because the SHIP statement does not require an option, however, you need not code this statement at all. In this situation, the default option output will be applied.

CLEAR Statements for Package Ship SCL

CLEAR statements clear 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 follow that SET statement. The CLEAR statement remains in effect until a new, related SET statement is encountered, or until processing ends.

CLEAR statements apply only to SET statements; similar information entered in a SHIP statement is not affected by the CLEAR statement.

Two CLEAR statements can be used with the Package Shipment Utility:

CLEAR DESTINATION

The CLEAR DESTINATION statement clears the destination specified in a previous SET statement.

CLEAR OPTIONS

The CLEAR OPTIONS statement clears the option specified in a previous SET statement.

How to Manage Package Ship

The CA Endevor SCM administrator can use the following tools to monitor and review the shipment of packages:

- The Package Ship reports (see page 175)
- Package Shipment assembler reports (see page 178)
- Shipment confirmation emails (see page 179)
- Display Shipment Status—The Display shipment status option, accessible from the Package Ship Destination panel, lets you view a scrollable list showing shipment status, including staging, remote copy, and script execution status. For more information, see Display Shipment Status (see page 171).

The Package Shipment Reports

The package shipment utility produces the following reports:

- Package Shipment Log—The Package Shipment Log Report contains a series of numbered messages which report the progress of the shipment staging process.
- Package Shipment Validation—The Package Shipment Validation report lists the "SHIP" transactions which were read as input by the staging utility. If a syntax error is detected, a numbered message follows the transaction. The final action (marked "FINISHED") tallies the number of actions processed, the number of actions in error, and the number of shipments to be made.
- Package Shipment Activity—The Package Shipment Activity report lists the data sets/members which were staged for shipment.
- Package Shipment Summary—The Package Shipment Summary report is broken down by destination and shows the staging return code and the packages shipped to each destination.

The Package Shipment Activity Report

The Package Shipment Activity report lists the data sets/members which were staged for shipment. The report is broken down by HOST data set name.

The header line shows the following three items of information:

- SHIPMENT OF MODULES or COMPLEMENTARY FILE; the former is the requested shipment and the latter is a backout of the shipment.
- The host data set name.
- The destination.

Detail lines follow the header. There is one detail line for each member in each package being shipped from the host data set to the destination mentioned in the header. If a member is in more than one package, it is listed multiple times. Each detail line contains the following information:

Action

Indicates how the shipment utility handles the member. Possible values are COPY, DELETE, and BYPASS.

COPY

Indicates that the member was copied to the host staging data set and will be shipped.

DELETE

Indicates that an IDCAMS DELETE command will be shipped to the remote site.

BYPASS

Indicates that the member is in multiple packages and this occurrence of the member is not being shipped.

Mbr-Name

The name of the HOST data set member.

Option

Indicates the shipment option. Possible values are OUTPUT and BACKOUT.

OUTPUT

Indicates that the member being shipped is part of the package output.

BACKOUT

Indicates that the member being shipped is the backout of the package member.

Package Name

The name of the package in which this member is a participant.

Associated Backout Data And Package Name

This column can contain an encrypted member name and or the package with which it is associated. The encrypted name represents an actual backout or backin member and is for the use of CA Endevor SCM Technical Support. Members are never shipped under the encrypted name, but rather under the name found in the MBR-NAME column.

Comments, Messages, Etc.

Contains a warning or error message for that detail line. It is prefixed by a return code.

RC=04 SUPERCEDED BY **<pkg-id>**

The package member associated with this detail line will not be shipped because it was superseded by the application of another package.

RC=12 OPTION SWITCH

The requested member cannot be shipped because it participates in two or more packages within this shipment and the options "OUTPUT" or "BACKOUT" are not the same. Break this shipment up into multiple shipments to remove this conflict.

RC=12 MEMBER NOT FOUND (ASSOCIATED MEMBER NOT FOUND)

The package member (or associated member) was not found in the HOST data set. It was probably deleted manually. Package integrity is lost.

RC=12 UNEXPECTED FORMAT COMBO (###)

This situation can occur when the member has participated in multiple packages and one of the packages in the hierarchy was deleted or reset and reused. The ### is a decimal representation of an internal format code.

RC=12 MCF DATA NOT FOUND

The MCF record for this member's element was not found or user is not authorized to access its environment.

RC=16 ERROR - FIND MEMBER ROUTINE (###)

An I/O error occurred while checking for the presence of a member to be shipped. Check the Shipment Log Report for more information. In addition, check the JES log for equipment or file problems. The ### is a return code.

RC=16 ERROR WRITING TO SYSIN FILE (###)

An I/O error occurred while writing a "SELECT MEMBER" card to the BSTCOPY SYSIN data set. Check the Shipment Log Report for more information. In addition, check the JES log for equipment or file problems. The ### is a return code.

Following all the detail lines for a given data set are eight trailer lines, which contain the following information:

- The first two show how many members were copied and how many will be deleted.
- The next four show the correspondence between output data set names and staging data set names at the host and remote site. If a DSNAME Mapping Rule was set up to exclude transmission of this data set, the "HOST LIBRARY DSN:" line will be marked "* DATASET EXCLUDED FROM TRANSMISSION *."
- The final two lines show the DSNAME Mapping Rule which was used to generate the "REMOTE LIBRARY DSN." If blank, the host data set name is used as the remote data set name.

Package Shipment Assembler Reports

Shipment reports allow you to review package shipment activity in summary format. To generate CA Endevor SCM Assembler reports, you must execute the BC1JRPTS job.For more information about requesting reports, see the *Reports Guide*.

Note: Shipment reports and Footprint reports are mutually exclusive. If you need both types of reports, you must submit two separate jobs or two separate jobsteps within the same job.

You can request the following shipment reports:

- CONRPT73—Destination Detail Report
- CONRPT74—Package Shipment Report by Package ID
- CONRPT75—Package Shipment Report by Destination
- CONRPT76—Package Shipment Report by Shipments

Shipment Confirmation Email Notification

The Shipment Confirmation Email Notification feature enables emails to be built and sent to the user that initiated the shipment request. The emails are sent during the shipment process when the host confirmation occurs and when the remote confirmation occurs. If shipments are sent to multiple destinations, separate emails are sent for each package and destination combination. The information contained in the emails is the same data shown when you display the Package Shipment Status panel.

The Shipment Confirmation Email Notification feature is in effect for any shipment request, whether initiated from CA CM Enterprise Workbench, an API program, or a CA Endevor SCM online Ship action.

This feature is implemented using the existing Email Notification Facility. The Email Notification Facility's Notification Facility table, ESMTPTBL, is shared with other features, such as override signout notification and package approval notification. However, you can disable the shipment confirmation feature by enabling the SUPPRESS_SHIPCONF_EMAIL option in the Optional Features Table, ENCOPTBL.

Note: For more information about the Email Notification facility, see the appendix "Email Notification" in the *Administration Guide*.

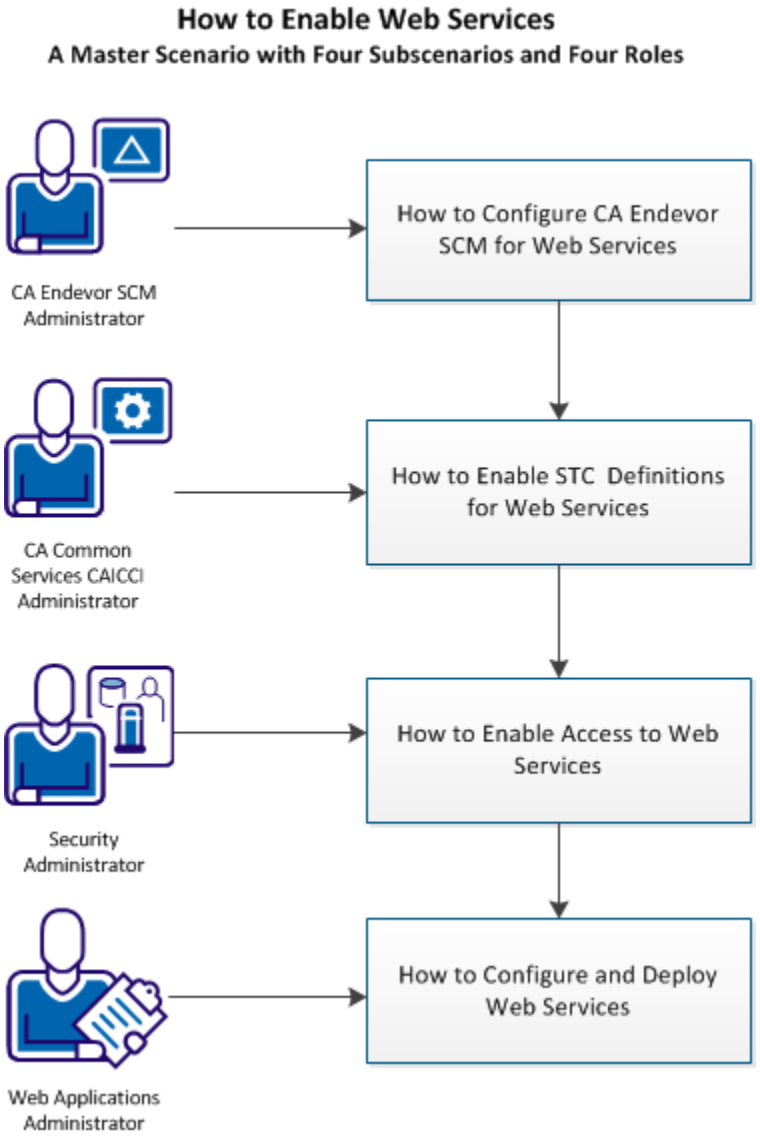
Chapter 6: Web Services

This section contains the following topics:

How to Enable Web Services

As a change manager (CA Endevor SCM administrator), you can enable the Web Services component. This component is a prerequisite for the Eclipse-Based UI. In addition, you can use Web Services to connect your user-written program to the CA Endevor SCM API.

Enabling Web Services is a complex process that requires that you coordinate the assistance of other roles. The following graphic shows the responsibilities (scenarios) associated with each role for the master scenario How to Enable Web Services:



To enable Web Services, the following roles complete these scenarios:

Note: All roles should review the prerequisites topics before attempting their how-to scenario.

1. CA Endevor SCM administrator— How to Configure CA Endevor SCM for Web Services (see page 187).
2. CA Common Services CAICCI administrator— How to Enable STC Definitions for Web Services (see page 195).
3. Security administrator— How to Enable Security Access to Web Services (see page 202).
4. Web applications administrator— How to Configure and Deploy Web Services (see page 208).

Note: Completing these steps sets up Web Services to use the Eclipse Plug-in. For more information about client stubs, see User-Written Client Programs (see page 228).

Review Prerequisite Information

Before enabling Web Services, make sure that you understand how Web Services works. Also review the software and setup required before deploying Web Services. Review the following topics:

- Web Services (see page 183)
- How Client Programs Access the API (see page 184)
- Software requirements for Web Services (see page 186)

Web Services

The CA Endevor SCM Web Services component enables client applications to communicate with the CA Endevor SCM API. Web Services runs in an HTTP Application server (Tomcat) under a z/OS USS environment. Two web service client design models are supported— SOAP and RESTful. For the SOAP model, Web Services uses Axis2 for SOAP message handling. Jersey (JAX-RS) is used for the RESTful model.

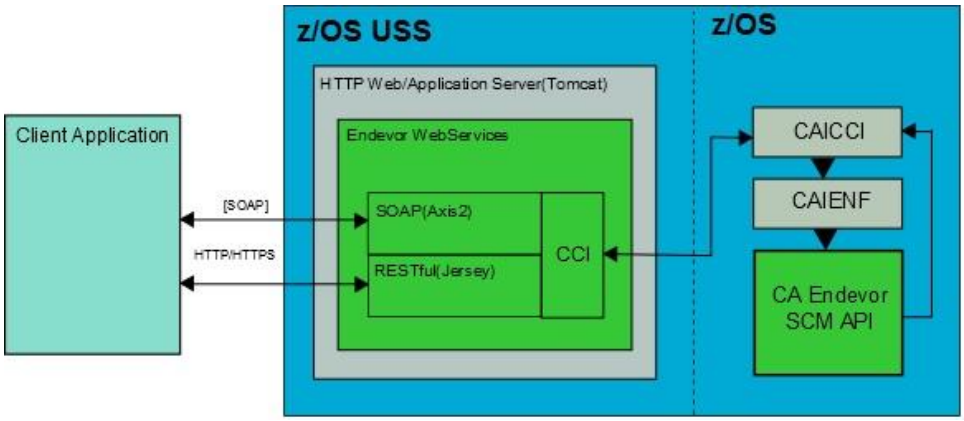
The Eclipse-Based UI and user-written programs use Web Services to send Software Control Language (SCL) statements, with some restrictions, (directly or indirectly, depending on the client model used) to the CA Endevor SCM API. Web Services lets you perform actions or queries on CA Endevor SCM abstractions, such as Elements, Packages, Types, Processors, and so on. You can use defined web service functions to perform Endevor actions and queries. Queries can return information to the web service caller in Comma Separate Value (CSV) format (using the submitSCL operation), in XML format (using Get<Object>) or in JSON format (using the RESTful API).

Possible client applications include thin or thick clients, browers, mobile application, or Rich Client Platform (RCP) programs.

How Client Programs Access the API

Web Services is a multithreaded application built with Java2 (the Apache Axis2/Java web services/SOAP/WSDL engine) and the Jersey framework (which serves as a JAX-RS reference implementation for RESTful web services). Web Services and client applications communicate using the Simple Object Access Protocol (SOAP) or the direct RESTful approach. The client and Web Services communicate in a request-response fashion using the HTTP protocol.

The following graphic shows the flow of information between the client, Web Services, and the CA Endevor SCM API.



As shown in the graphic, client applications access the CA Endevor SCM API through Web Services. Details of this process follow:

- If the SOAP web service client model is used, then the client application formats and sends a SOAP message to the HTTP/Application server based on the definitions in the Web Services Description Language (WSDL) file. The request includes the name of the CA Endevor SCM data source that the client wants to access.
- With the RESTful model, the client creates an HTTP request that includes a specific URL and parameters. The request corresponds to a specific Endevor action and sets the appropriate HTTP method associated with the action.
- Web Services matches the name of the data source requested by the client call to a configuration file. One or more configuration files are stored on the HTTP web server and enable access to the CA Endevor SCM API. A configuration file identifies the name and number of API STCs (the STC pool) available to pass requests to CA Endevor SCM. The CA Common Services CAICCI Spawn facility spawns the specified number of API STCs.

- CA Endevor SCM processes the request and then the results are returned by CAICCI to Web Services where the data are processed and transformed into the appropriate response structure and then returned to the client.

  To enable Web Services to process requests and communicate with the API, the z/OS environment where CA Endevor SCM is installed requires certain CA Common Services components, including the following:

  - CAICCI provides a common communications software layer that enables communication between Web Services and the API.
  - CAIENF must be configured to enable the CAICCI Spawn facility that lets CA Endevor SCM schedule, execute, and monitor a pool of STCs for Web Services requests.

This process uses the following standard technologies:

- **XML language –** The data format used by web service components. Communications between web service applications are written in XML format.
- **Simple Object Access Protocol (SOAP) –** An XML-based messaging protocol and encoding format for inter-application communication. SOAP messages are XML messages sent between applications. A SOAP message is a text file written in standard XML format that is enclosed in a SOAP structure. SOAP enables applications to know how to send and interpret the XML data.
- **Web Services Description Language (WSDL) –** A service description protocol. A web service's WSDL file is a dynamic XML file that serves as the interface between a SOAP-based client program and the web service. The Web Service WSDL file describes what CA Endevor SCM operations are available to the client program and the SOAP protocol bindings and message formats required to enable the client to interact with the web service.

  If you are using Web Services for a user-written client application, your web developer must create a client stub from the WSDL file. The client stub is responsible for conversion of parameters used in a function call and deconversion of results passed from the server after execution of the function. The client stub interacts with the web service stub. For more information see, CA Endevor SCM SOAP Web Services (see page 228).
- **Web Application Description Language (WADL)—** Jersey supports WADL. WADL is an XML description of the deployed RESTful web application. It contains the model of the resources, structure, supported media types, HTTP methods, and so on. It is similar to the WSDL, which is used for the SOAP-based Web Services model. WADL can be used for generating a client-side stub, however this feature is not as sophisticated as the stubs generated using the WSDL.
- **HTTP Protocol—** The Hyper Text Transfer protocol (HTTP) is an application protocol for distributed, collaborative information systems, and is the foundation for data communication on the World Wide Web.

- **JSON (JavaScript Object Notation)—** JSON is an open standard, language independent, data format used to transmit data objects as structured attribute-value pairs in human-readable text. It is considered an alternative to XML.
- **JAX-RS—** Java API for RESTful Services (JAX-RS) is a Java programing language API that provides support for the Representational State Transfer (REST) type of web services.

Software Requirements for Web Services

The following mainframe software is required:

- CA Endevor SCM Version 17.0.00
- CA Common Services:
  - If you are using CA Common Services Version 14.0, verify that the following PTFs are applied:
    - PTF RO50139. This PTF affects CAICCI.
    - PTF RO47492. This PTF is required to make VSAM Extended Addressability available through CA L-Serv. (This PTF has been required beginning with CA Endevor SCM Release 15.1. For more information about CA L-Serv support of VSAM EA, see the *CA Endevor SCM Release 15.1 Release Notes*.)
  - If you are using CA Common Services Release 14.1, verify that the following PTF is applied:
    - PTF RO52401. This PTF affects CAICCI.
- Java 7
- Apache Tomcat 7.0 for the HTTP and application server.

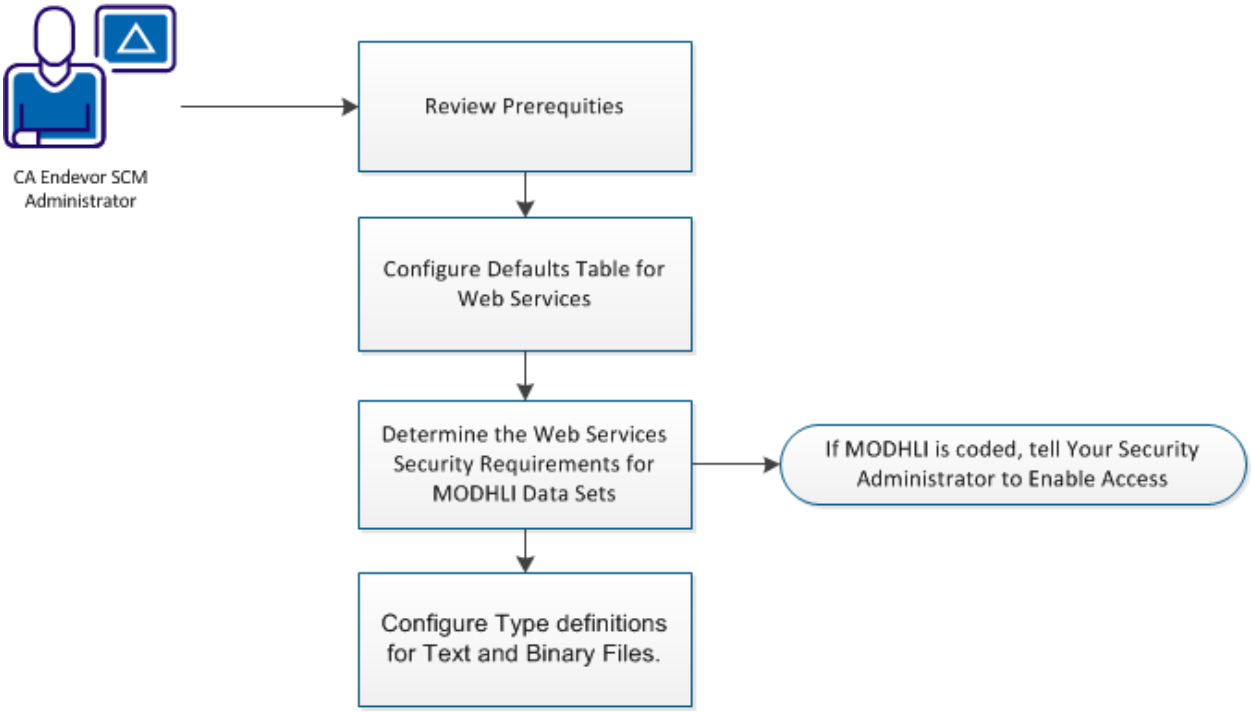
  You can install a supported version of Apache Tomcat from CA Common Services for z/OS. For more information on this process, see the *CA Common Services for z/OS Installation Guide*.
- Apache Axis2/Java 1.6 and Jersey 2.7 frameworks.

  You can set/install these components when you perform the procedure "How to Configure and Deploy Web Services (see page 208)."

How to Configure CA Endevor SCM for Web Services

As a change manager (CA Endevor SCM administrator), you can configure CA Endevor SCM to support the Web Services component. This component is a prerequisite for the Eclipse-Based UI. In addition, you can use Web Services to connect your user-written program to the CA Endevor SCM API.

How to Configure CA Endevor SCM for Web Services



To configure CA Endevor SCM, do the following:

1. Configure the Defaults table for Web Services (see page 188).
2. Determine the Web Services security requirements for MODHLI Data Sets (see page 189).
3. Configure Type definitions for Text and Binary Files. Review the information in Support for Element Types (see page 189) topics. Then define Type definitions as appropriate for your site as described in the following topics:
   - How to Define an Element Type for Text Files (see page 191)
   - How to Define an Element Type for Binary Files (see page 192)
4. (Optional) To optimize the use of your Master Control File (MCF) and Package data set, we strongly recommend that you implement IBM VSAM Record Level Sharing Support (RLS). For more information, see the appropriate documentation provided by IBM.

Configure Defaults Table for Web Services

To enable CA Endevor SCM to use Web Services, you must customize the TYPE=MAIN section of the C1DEFLTS table (Defaults table).

Follow these steps:

1. Set the BATCHID parameter in the TYPE=MAIN section of the C1DEFLTS table to 1. This specifies that the CA Endevor SCM user ID associated with a batch job will be determined by the USER parameter specified on the JOB statement instead of the JOBNAME.

2. Specify a value for MODHLI, *if* your site uses RACF with the RACF PROTECTALL enabled. When using the Eclipse-Based UI, CA CMEW, or any other client program under RACF with the RACF PROTECTALL option activated, security violations are issued, unless MODHLI is coded. This occurs because the ID used to allocate the temporary files required by Web Services is different than the ID used to open the files. Specifically, the data sets are allocated under the context of the user ID provided on the CA Common Services Common Communications Interface (CAICCI) request to spawn the pool STCs. The open and write requests are issued under the context of the user ID signed on to the client program that is issuing requests through Web Services.

   Note: MODHLI affects security. For more information, see Determine the Web Services Security Requirements for MODHLI (see page 189).

3. Assemble and link-edit the table. You can use an SMP/E USERMOD to assemble and link-edit C1DEFLTS after it has been customized. Alternatively, you can edit the sample JCL BC1JTABL to assemble and link source module C1DEFLTS outside of SMP/E. BC1JTABL is supplied in the installation library iprfx.iqual.CSIQJCL. This stores the defaults table in iprfx.iqual.CSIQAUTU as member C1DEFLTS.

Note: For more information about editing the Defaults table, see the *CA Endevor Software Change Manager Administration Guide*.

Determine the Web Services Security Requirements for MODHLI Data Sets

To enable client programs to access the API, your security administrator may need to enable user access to MODHLI data sets. The CA Endevor SCM administrator must determine the security requirements for MODHLI data sets.

Follow these steps:

1. View your C1DEFLTS table, to see if a MODHLI value is coded in the TYPE=MAIN section of the table.
   - If the MODHLI value is coded, your security administrator must grant all user IDs access to all data sets with the MODHLI high-level qualifier (HLQ). This requirement applies to access under all security products including RACF, CA Top Secret, or CA ACF2.

     The reason for this requirement is as follows. For a client program to access the API, the user ID sent to CAICCI to spawn the pool STCs and the user IDs that issue requests to Web Services must have read/write access to these data sets. To enable this, the MODHLI parameter causes the data set names to be built with this format:

     modhli.D*yyddd*.T*hhmmss*.STC*nnnnn*.*ddname*

     STCnnnnn, ddname

     The job ID and ddname that is the unique qualifier for one of the nine API related files (APIMSGS, C1MSGS1, and so on).
   - If a MODLHI value is not coded, then security is not affected, because the temporary data sets names are built by the operating system with the standard temporary data set HLQ in the following format: SYSyyddd.Thhmmss.RA000.jobname.nnnnnn
2. Tell your security administrator to grant all user IDs access to all data sets with the MODHLI high-level qualifier (HLQ), if the MODHLI value is coded in C1DEFLTS.

Support for Element Types

You can add files accessible on a client machine to CA Endevor SCM and retrieve elements from CA Endevor SCM. Web Services transfers files to and from the client application using SOAP attachments. To enable the translation of transferred data, a CA Endevor SCM Type record must be properly defined for each type of data to be managed. The characteristics of binary and text files are described next:

Text files

CA Endevor SCM defines a text file as one that has line delimiters and requires character translation. You can store text files on the mainframe and browse them in CA Endevor SCM. Detailed change history is stored for text files.

Browsing an element in CA Endevor SCM invokes the EBCDIC-to-ASCII conversion function of CA Endevor SCM; the EBCDIC-to-local file system encoding is not used. However, when you retrieve a text file from CA Endevor SCM, the Web Services server converts EBCDIC from the mainframe.

Important: The CA Endevor SCM administrator must work with the web services administrator to properly define the element Type for text files.

The CodePage Web Services parameter in the Web Services configuration file defines the character sets for the mainframe and Web Services. The configuration file is customized by your Web Services administrator. A description of this parameter follows:

CodePage

Specifies the code page, defined in the mainframe operating system, that CA Endevor SCM is using. The code page is a variation of the Extended Binary Coded Decimal Interchange Code (EBCDIC) that defines how text files are coded on the mainframe.

The CodePage and CharacterSet parameters enable Web Services to translate the plain text information exchanged between the client computer and the API. Plain text information includes SCL commands sent to the API and reports and CSV files extracted from the API.

For information about supported encodings, see: http://download.oracle.com/javase/1.5.0/docs/guide/intl/encoding.doc.html

Default: cp01140

Note: For more information about the configuration file parameters, see ENDEVOR.cfg Parameters (see page 221).

Binary files

CA Endevor SCM defines a binary file as one that does not have line delimiters or require character translation. Examples of binary files are Microsoft Word documents or Excel spreadsheets. Long names are typically used for naming binary files. Specific file versions can be retrieved through Web Services. You can browse binary files whose content type is recognized by your browser when using Web Services. Detailed change history is stored for binary files, except for file Types defined with log delta format. For more information about binary files, see How to Define an Element Type for Binary Files.

How to Define an Element Type for Text Files

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

1. Define the base and delta libraries using CA Endevor SCM rules for the specific file type. For more information, see the *CA Endevor Software Change Manager Administration Guide*.
2. Set TYPE DEFINITION panel fields to the following values to define a CA Endevor SCM Type record for text data.
    a. Set the FWD/REV/IMG DELTA field to F for forward or R for reverse to specify the delta storage format.
    b. Set the COMPRESS BASE/ENCRYPT NAME field to N for element names that do not exceed eight characters and are uppercase. For other element names, set this field to Y. If set to Y, the base library must be HFS.
    c. Set the COMPARE FROM field to a value appropriate for the type of data. For example, COBOL can be 7.
    d. Set the SOURCE LENGTH and COMPARE TO fields to values appropriate for the type of data. For example, COBOL SOURCE LENGTH can be 80 and COMPARE TO can be 72.
    e. Set the HFS RECFM field to CRLF. This is the line feed character for Microsoft Windows files. If you do not specify CRLF for text types accessed using local file system support, you will get unexpected results in Windows-based editors.
    f. Set the DATA FORMAT field to T for text. This triggers conversion from EBCDIC to local file system encoding.
    g. Set the FILE EXT field to the file name extension you want the file type to have on the local machine (cob, jcl, and so on) or leave it blank. FILE EXT is used when you are adding files from or retrieving files to a local directory using Web Services.

How to Define an Element Type for Binary Files

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

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

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

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

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

How to Resize a zFS Linear Data Set

Web Services is installed as a linear VSAM file, *iprfx.iqual*.CSIQUSSM, which contains a file system that is mounted to a user-defined mount point on the Z/OS UNIX file system.

A user can receive the error message “EDC5133I No space left on device”, which indicates that there is no free space left in the z/OS UNIX file system. As a Web Services administrator, you can allocate additional space for the file system.

A sample batch JCL job, WSZFSRSZ, is delivered in iprfx.iqual.CSIQJCL. This job, in summary, does the following:

1. Deletes work files
2. Produces details of the original zFS linear data set into the job output
3. Unmounts the original zFS linear data set from the z/OS UNIX file system
4. Allocates a new zFS linear data set with your allocation values
5. Creates a backup data set with the data from the original zFS data set
6. Alters the name of the original zFS linear data set to ‘zFSFile.OLD’
7. Alters the name of the new zFS linear data set to the original zFS linear data set name
8. Restores the data from the backup data set into the resized zFS linear data set

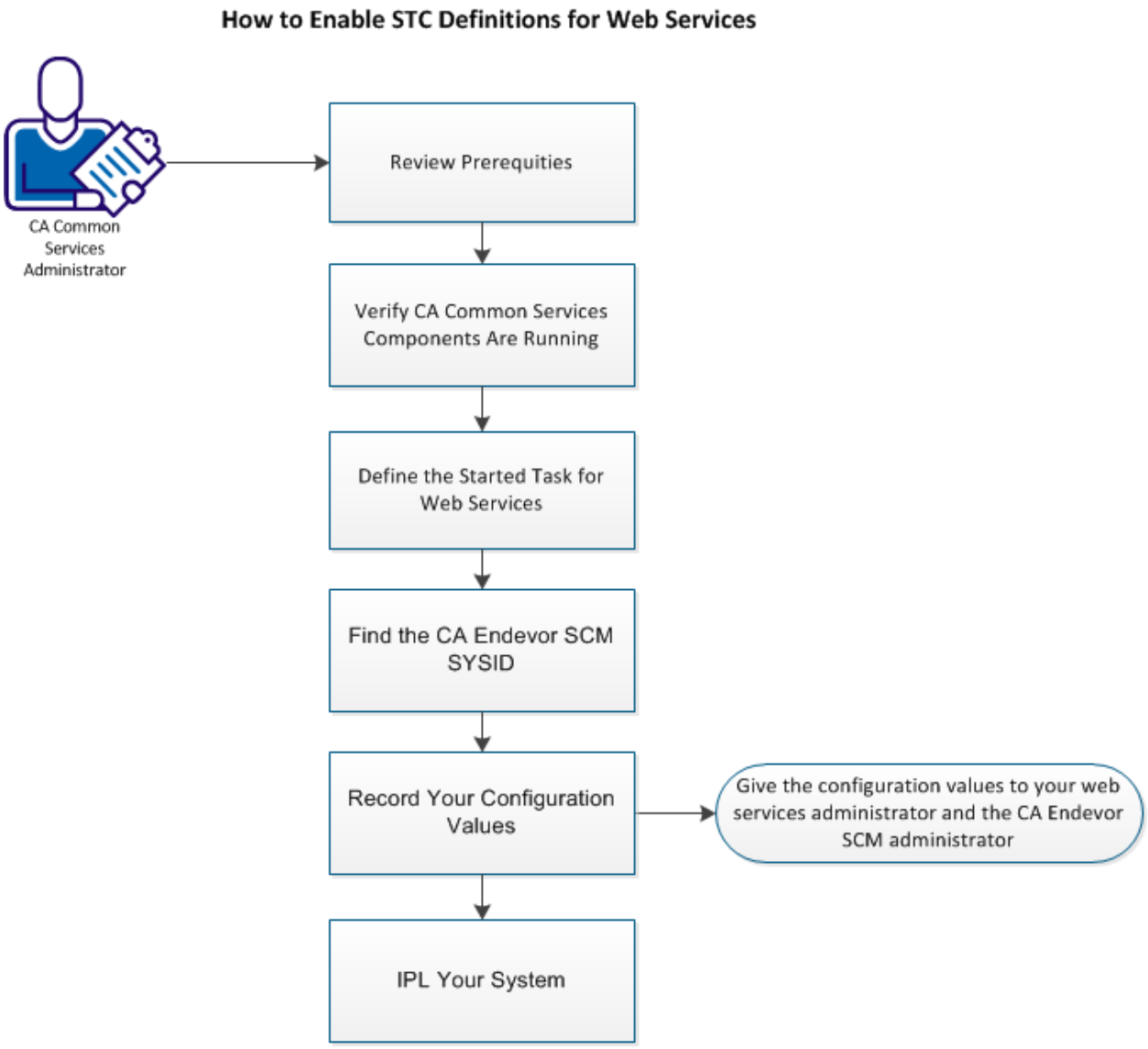
9. Produces details of the resized zFS linear data set into the job output
10. Mounts the resized zFS linear data set to the z/OS UNIX file system

After the successful execution of the batch JCL job, WSZFSRSZ, you can view the job output or view the VSAM attributes to verify the data set was resized correctly.

How to Enable STC Definitions for Web Services

As a CA Common Services administrator, you can configure the CA Common Services components to support Web Services for CA Endevor SCM. Web Services is a prerequisite for the Eclipse-Based UI and user-written programs that connect to the CA Endevor SCM API. To enable communication between CA Endevor SCM and Web Services, certain CA Common Services components must be installed and configured.

The following graphic shows how you can enable STC definitions for Web Services:



To enable STC definitions for Web Services, do the following:

1. Review the prerequisites (see page 183) for an understanding of how Web Services works.
2. Verify that the CA Common Services components CAIRIM, CAIENF, and CAICCI are installed and running (see page 196) on the mainframe. If your site is running CAIENF r12 or higher, a database is not required for Web Services.
3. Modify the CAICCI Spawn facility to define the started task for Web Services. (see page 197)
4. Find the SYSID for the location where CA Endevor SCM is installed (see page 199).
5. Record the configuration values (see page 200) and give this information to your web services administrator.
6. IPL your system, if necessary. If this is your initial installation of CAICCI components, you do not need to IPL your system after installation. If this is a subsequent installation, you may need to IPL your system. For more information, see the *CA Common Services for z/OS Installation Guide*.

Verify that CA Common Services Components Are Running

The CCS components CAIRIM, CAIENF, and CAICCI must be installed and running on your mainframe. If they are not, you must install these services according to the instructions in the *CA Common Services for z/OS Installation* guide.

To verify whether CAIENF and CAICCI are installed and running

1. Go to the z/OS System Display and Search Facility (SDSF) status display and enter the prefix ENF*.
   - If you do not see ENF* active on your system, contact your systems programmer to start it.
   - If the ENF job is running, go to step 2.
2. Select the ENF job and search for (Find) CAICCI within the job.
   - If you find CAICCI, then CAIENF and CAICCI are running. You are done verifying that the CCS components are running. Go to the procedure How to Enable the Started Task for Web Services.
   - If you do *not* find CAICCI, then go to step 3.

3. Check your site's CAI.CAIPROC library to see if the ENF member has been modified to run CAICCI at your site.
   - If the ENF member exists, go to the task Customize the ENF Procedure and then have the member started.
   - If the ENF member does *not* exist, contact your local systems support person to determine the status of CAIRIM, CAIENF, and CAICCI on your system and to help you complete the tasks in the next section.

Define the Started Task for Web Services

Web Services uses started tasks that are started by the CAICCI SPAWN facility to communicate with the CA Endevor SCM API. You define these tasks to the spawn facility by adding a spawn parameter DD statement (*ndvspawn*) to the ENF procedure for CAICCI services. The *ndvspawn* DD statement specifies the name of the started task JCL (based on WSEWSSTC in CSIQJCL) for the CA Endevor SCM API. The WSEWSSTC procedure JCL must be edited to specify the high-level and second-level qualifiers for the CA Endevor SCM data sets. The WSEWSSTC copy is initiated as a started task.

Follow these steps:

1. Customize the ENF Procedure by adding the Web Services spawn parameter file name to the SPNPARMS DD in the ENF PROC JCL. You can find the ENF procedure in the initial CAI.CAIPROC library. However, your system administrator may have copied the procedure to the system PROCLIB. The following JCL is shown as an example to indicate the two lines you must add to the JCL.

```
//ENF PROC OPTLIB='SYS2.CA90S.PARMLIB',
//                  ENFDB='SYS2.CA31.ENFDB',
//                  ENFPARM=ENFPRM31,
//                  SPNPAR1=SPWNSNMP,
//                  SPNPARn=ndvspawn,
//                  CCIPARM=CCIPCA31,
//                  ENFCMDS=ENFCMD31,
//ENF      EXEC PGM=CAS9MNGR,TIME=1440
//CASRT01  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//CASRT02  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//CASRT03  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//CASRT04  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//CASRT05  DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//SPNPARMS DD DISP=SHR,DSN=&OPTLIB(&SPNPAR1)
//         DD DISP=SHR,DSN=&OPTLIB(&SPNPARn)
//SPNPRINT DD SYSOUT=X
//SPNDEBUG DD SYSOUT=X
//SRVDEBUG DD SYSOUT=X
//ENFDB DD DISP=SHR,DSN=&ENFDB
```

```
//ENFPARMS DD DISP=SHR,DSN=&OPTLIB(&ENFPARM)
//         DD DISP=SHR,DSN=&OPTLIB(&CCIPARM)
//ENFCMDS  DD DISP=SHR,DSN=&OPTLIB(&ENFCMDS)
//SYSPRINT DD SYSOUT=X
```

SPNPAR*n*

Specifies a symbolic for a CAIENF parameter member. Change *n* to the next available sequential number in the list of parameter files.

ndvspawn

Specifies the spawn parameter file. This member contains the CAICCI SERVICE and PROCESS statements. The SERVICE statement identifies the CA Endevor SCM API. The PROCESS statement identifies the JCL procedure that executes the CA Endevor SCM API.

2. Customize the spawn parameters by editing the CAICCI SERVICE and PROCESS statements in the *ndvspawn* file. Then save the appropriate file in your site specific CA90S.PARMLIB as member NDVSPAWN.

Important! For sample CCI definitions for Web Services, see the member WSEWSCCI supplied in the installation source library iprfx.iqual.CSIQOPTN.

SERVICE statement

Identifies the CA Endevor SCM API (host application).

PROCESS statement

Identifies the JCL procedure that executes the CA Endevor SCM API.

PROCNAME=*WSEWSSTC*

Specifies the name of the procedure associated with the CA Endevor SCM API host application that is started by the CAICCI SPAWN service during Web Services processing. The *WSEWSSTC* value is the name of the started task procedure that is initiated by a SPAWN request.

Note: By default, the name of the procedure associated with this started task is WSEWSSTC; however, this name is often changed, for example, to meet site-specific naming or security standards. If you do not know the procedure name at your site, check to see if the default name was retained or consult your system administrator.

Important! Keep the statements in the exact format provided, maintaining spaces and column alignments. For more information about defining and using CAICCI statements, see the CA Common Services documentation.

3. Customize the WSEWSSTC started task procedure. Change the parameters as appropriate for your site.

   Note: A sample WSEWSSTC procedure is located in the CSIQJCL library delivered with CA Endevor SCM.

   Note: If you change the WSEWSSTC member name, be sure to update the PROCNAME value in the CCI PROCESS statement of the SPNPARn file to reflect the new procedure name.

   Note: The wsewsstc value is specified in the JOBNAME parameter of the configuration file.

4. Copy WSEWSSTC into a PROCLIB defined to JES. WSEWSSTC is initiated as a started task.

5. If this is your initial install of these CAICCI components, you do not need to IPL your system after installation. If this is a subsequent install, you may need to IPL your system. For more information, see the CA Common Services for z/OS documentation.

Find the CA Endevor SCM SYSID

You must know the SYSID of the system where CA Endevor SCM is installed. Later you will need to enter the SYSID in the configuration file. For more information, see How to Create a Configuration File (see page 219).

To find the CA Endevor SCM SYSID, open the CCIPARM file member located in the library &OPTLIB. Find and make a note of the SYSID parameter.

For example, part of a sample CCIPARM file is shown next. In this sample, the name of the SYSID is A31SENF.

```
000001 ********************************************
000002 ** CA EVENT NOTIFICATION/CCI PARAMETER FILE *
000003 ** ---------------------- *
000004 ** CA-ENF/CCI VERSION 1.1 CONTROL OPTIONS *
000005 ********************************************
000006 SYSID(A31SENF)
000007 PROTOCOL(LU0,A31SENF,01,A31SENF)
000008 PROTOCOL(XES,CCIXES)
000009 PROTOCOL(TCPIP,1202)
```

Record Your Configuration Values

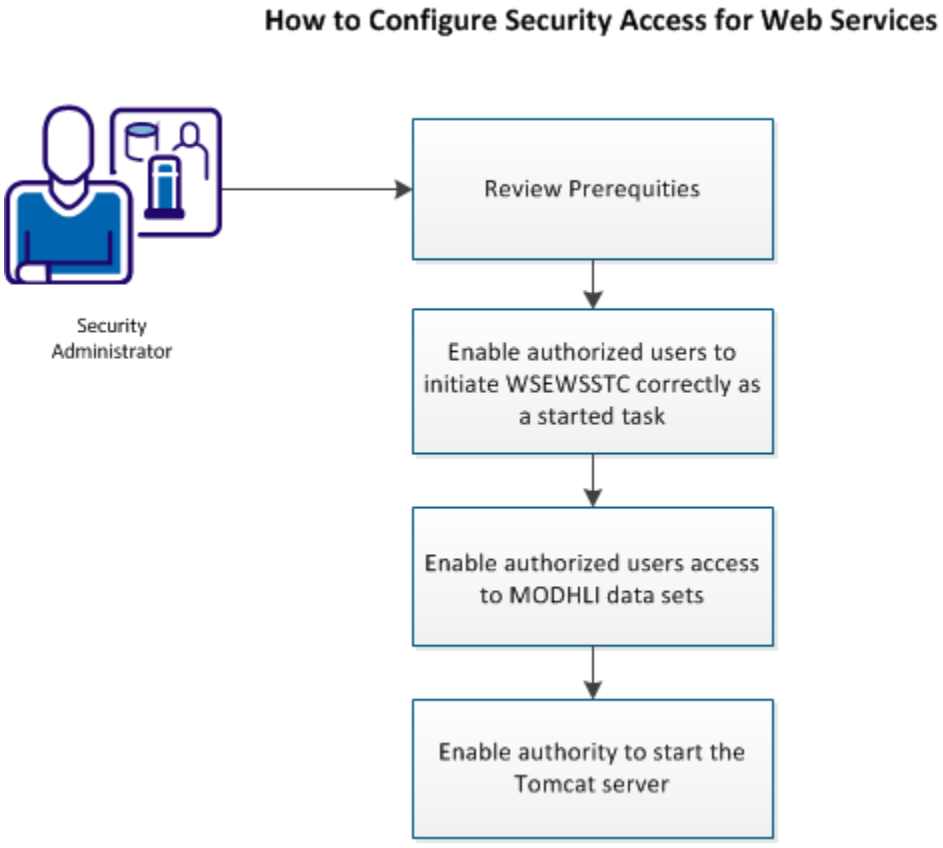
Record the values you identified while completing the procedures How to Enable STC Definitions for Web Services and Find the CA Endevor SCM SYSID. Your web services administrator will use some of these values to populate the configuration file. Record the following values:

Definition	Source	Default Value	Your Value
Mainframe logon host name: Specifies the name of the SYSID parameter in your CCIPARM file. This is also used for the configuration HostName parameter.	CCIPARM	none	
CA Endevor SCM mainframe started task procedure: Specifies the procedure associated with the started task that starts CA Endevor SCM services when Web Services starts. The default name for this procedure (WSEWSSTC) may have been changed to meet site-specific standards. This is also used for the configuration file's JobName parameter.	NDVSPAWN file	WSEWSSTC	
SPNPARn: Specifies a symbolic for a CAIENF parameter data set.	Systems administrator	SPNPAR2	

How to Enable Security Access to Web Services

As a security administrator, you can configure your site's security software to support Web Services for CA Endevor SCM. Web Services is a prerequisite for the Eclipse-Based UI and user-written programs that connect to the CA Endevor SCM API.

The following graphic shows how to enable security access for Web Services:



To enable security access for Web Services, do the following to configure your site's security software:

1. Enable authorized users to initiate WSEWSSTC correctly as a started task (see page 202).
2. Enable authorized users access to MODHLI data sets (see page 205), if MODHLI is defined in the CA Endevor SCM C1DEFLTS table.
3. Enable authority to start the Tomcat server (see page 206).

Security Software Customization for Web Services

You must configure your site's security software so that all authorized users can initiate WSEWSSTC correctly as a started task.

Note: Web Services uses started tasks that are started by the CA Common Services CAICCI SPAWN facility to communicate with the CA Endevor SCM API. Your site's CA Common Services administrator must configure the spawn facility for Web Services. The administrator configures the WSEWSSTC procedure to specify the high-level and second-level qualifiers for the CA Endevor SCM data sets. The WSEWSSTC copy is initiated as a started task.

Review the following sample configuration for the software used at your site. These samples are only examples and may not conform to your site's security standards.

- CA Top Secret (see page 202)
- CA ACF2 (see page 203)
- IBM RACF (see page 205)

For more information about setting up started task security, see the product-specific documentation or contact Technical Support for the security software you are using.

Note: Files are compressed, but no data encryption occurs during the transfer of files from the client to the Web Services server. Data communicated between CA Endevor SCM and Web Services is not compressed or encrypted, except for passwords.

Configure CA Top Secret

If your site uses CA Top Secret, you must configure CA Top Secret so that all authorized users can initiate WSEWSSTC correctly as a started task. To configure CA Top Secret, you define a new facility named ENDEVOR, add a new ACID, and give the ACID a MASTFAC definition. You then make this facility the default ACID for the CAICCI-spawned task WSEWSSTC defined by your site's CA Common Services administrator. Finally, you add all users of Web Services to the ENDEVOR facility.

Note: For more information about completing the following steps, see your CA Top Secret documentation or contact Technical Support for CA Top Secret.

To configure CA Top Secret

1. Define a new facility named ENDEVOR for CA Endevor SCM by adding the following definitions to the CA Top Secret parameter file (specified by the PARMFIELD DD statement):

```
* USERnn FACILITY FOR
*
FAC(USERnn=NAME=ENDEVOR)
```

2. Define a new ACID named ENDEVOR by entering the following command:

   ```
   TSS CRE(ENDEVOR) NAME('endevor userid') TYPE(USER) FAC(STC,ENDEVOR) PAS(NOPW,0)
   ```

   The NODSNCHK, NORESCHK, and NOSUBCHK bypass attributes on the ENDEVOR ACID may be required. If not, make sure that the ACID is authorized to access all of the files and resources it requires.

3. Give the ENDEVOR ACID a MASTFAC definition by entering the following command:

   ```
   TSS ADD(ENDEVOR) MASTFAC(ENDEVOR)
   ```

4. Assign ENDEVOR as the default ACID for the CAICCI-spawned task WSEWSSTC by entering the following command:

   ```
   TSS ADD(STC) PROCNAME(WSEWSSTC) ACID(ENDEVOR)
   ```

5. Grant each user of Web Services access to the ENDEVOR facility by entering the following command:

   ```
   TSS ADD(USERID) FAC(ENDEVOR)
   ```

Note: For more information about defining a new facility, see your *CA Top Secret Control Options Guide*. For more information about the CRE and ADD commands, see the *CA Top Secret Command Functions Guide*.

Configure CA ACF2

If your site uses CA ACF2, you must configure CA ACF2 so that all authorized users can initiate WSEWSSTC correctly as a started task. To configure CA ACF2, you create an STC login, verify that the dataset access is correct, and define a resource class. This section is only a sample of how your site can set up security. For more information about setting up started task security, see your CA ACF2 documentation or contact Technical Support for CA ACF2.

To configure CA ACF2

1. Create an STC logon ID named ENDEVOR for the WSEWSSTC started task by entering the following commands:

   ```
   ACF
   INSERT ENDEVOR NAME(ENDEVOR) STC
   ```

2. Verify that the ENDEVOR logon ID is defined with site-specific logon ID fields such as those fields used to create the UID string.

   Note: For instructions to create CA ACF2 logon ID records, see the *CA ACF2 Administration Guide*.

3. Verify that the ENDEVOR logon ID has access to all required datasets by writing CA ACF2 ACCESS rules for the logon ID.

   Note: For instructions to write these rules, see the *CA ACF2 Administration Guide*.

4. Define a resource class called FACILITY and assign a resource type code of FAC. Perform these steps to do so:

   a. Enter the following commands to create the CLASMAP record for the

      ```
      FACILITY resource class:
      ACF
      SET CONTROL(GSO)
      INSERT CLASMAP.FAC RESOURCE(FACILITY) RSRCTYP(FAC)
      ```

   b. Enter the following commands to add the FAC resource type code to the CA-ACF2 GSO INFODIR record:

      ```
      SET CONTROL(GSO)
      CHANGE INFODIR TYPES(R-RFAC)
      ```

   c. Do one of the following to activate the CLASMAP and the INFODIR record change:

      - Restart the CA-ACF2 address space.
      - Enter the following commands:

        ```
        F ACF2,REFRESH(CLASMAP)
        F ACF2,REFRESH(INFODIR)
        ```

   Note: For more information about maintaining CA-ACF2 GSO records, see the *CA ACF2 Administration Guide*.

5. Create a FACILITY resource rule record called ENDEVOR and grant users access to this resource by issuing the following commands:

   ```
   ACF
   SET RESOURCE(FAC)
   COMPILE */pds.name
   $KEY(ENDEVOR) TYPE(FAC)
   UID(user1 uid string) ALLOW
   UID(user2 uid string) ALLOW
   .......
   STORE
   ```

6. Enter the following command to rebuild the FAC directory:

   ```
   F ACF2,REBUILD(FAC)
   ```

   Note: For instructions to write CA ACF2 resource rules, see the *CA ACF2 Administration Guide*.

Configure IBM RACF

This section provides basic instructions for customizing IBM RACF to allow the WSEWSSTC started task to initialize correctly. According to the *RACF Security Administrator Guide*, you can use either of the following methods to define a started task to RACF:

- Define a new profile to the STARTED class (recommended by IBM)
- Add a new entry in the started procedures table (ICHRIN03)

Additionally, you must assign a RACF user ID to the started task WSEWSSTC and assign the user ID to a RACF group authorized to initiate started procedures.

To define a RACF user ID for WSEWSSTC, use the ADDUSER command and associate it with your existing started task RACF group, as follows:

```
ADDUSER user_name DFLTGRP(default_group) OWNER(default_group) NOPASSWORD
```

user_name

Specifies the name of the new RACF user ID. This name should be the same as the name of the started task member in your PROCLIB that Web Services uses.

default_group

Specifies the default group that contains all system started tasks; for example, STCGROUP.

Note: If you do not know the name of the default group, see your RACF administrator. For detailed information to implement the RACF STARTED class or to modify the started task table (ICHRIN03), see the RACF documentation.

Note: This command is only an example. For more information on using the ADDUSER command, see your RACF administrator.

Enable User Access to MODHLI Data Sets

To enable client programs to access the API, your security administrator may need to enable user access to MODHLI data sets. The CA Endevor SCM administrator must determine the security requirements for MODHLI data sets.

Follow these steps:

1. Ask your CA Endevor SCM administrator if MODHLI is coded in the CA Endevor SCM C1DEFLTS table.
   - If the MODHLI value is coded, you must grant all user IDs access to all data sets with the MODHLI high-level qualifier (HLQ). This requirement applies to access under all security products including RACF, CA Top Secret, or CA ACF2.

     The reason for this requirement is as follows. For a client program to access the API, the user ID sent to CAICCI to spawn the pool STCs and the user IDs that issue requests to Web Services must have read/write access to these data sets. To enable this, the MODHLI parameter causes the data set names to be built with this format:

     modhli.D*yyddd*.T*hhmmss*.STC*nnnnn*.*ddname*

     STCnnnnn, ddname

     The job ID and ddname that is the unique qualifier for one of the nine API related files (APIMSGS, C1MSGS1, and so on).
   - If a MODLHI value is not coded, then security is not affected, because the temporary data sets names are built by the operating system with the standard temporary data set HLQ in the following format: SYSyyddd.Thhmmss.RA000.jobname.nnnnnn
2. Configure your security software to grant all user IDs access to all data sets with the MODHLI high-level qualifier (HLQ), if the MODHLI value is coded in C1DEFLTS.

Enable Authority to Start the Tomcat Server

To enable an administrator to start the Tomcat server, their user ID must have access to certain JVM environment values used by the Tomcat Server. These values are set by your web services administrator. The security administrator grants access to these paths in your site's security software.

Follow these steps:

1. Ask your web services administrator what values are set for the following JVM environment values. The web services administrator sets these values in member WSTOMENV.

   INSTALL_HOME— Specifies the location where Tomcat has been installed on the system. This should be based on the value of CCS_TOMCAT_DIR.

   PRODUCT_HOME— Specifies the location where Web Services will be implemented. This should be based on the value of TARGET_TOMCAT_DIR.

2. Update your security software to enable the appropriate users to access the paths as follows:
   - Read access to the path defined in INSTALL_HOME.
   - Write access to the path defined in PRODUCT_HOME.

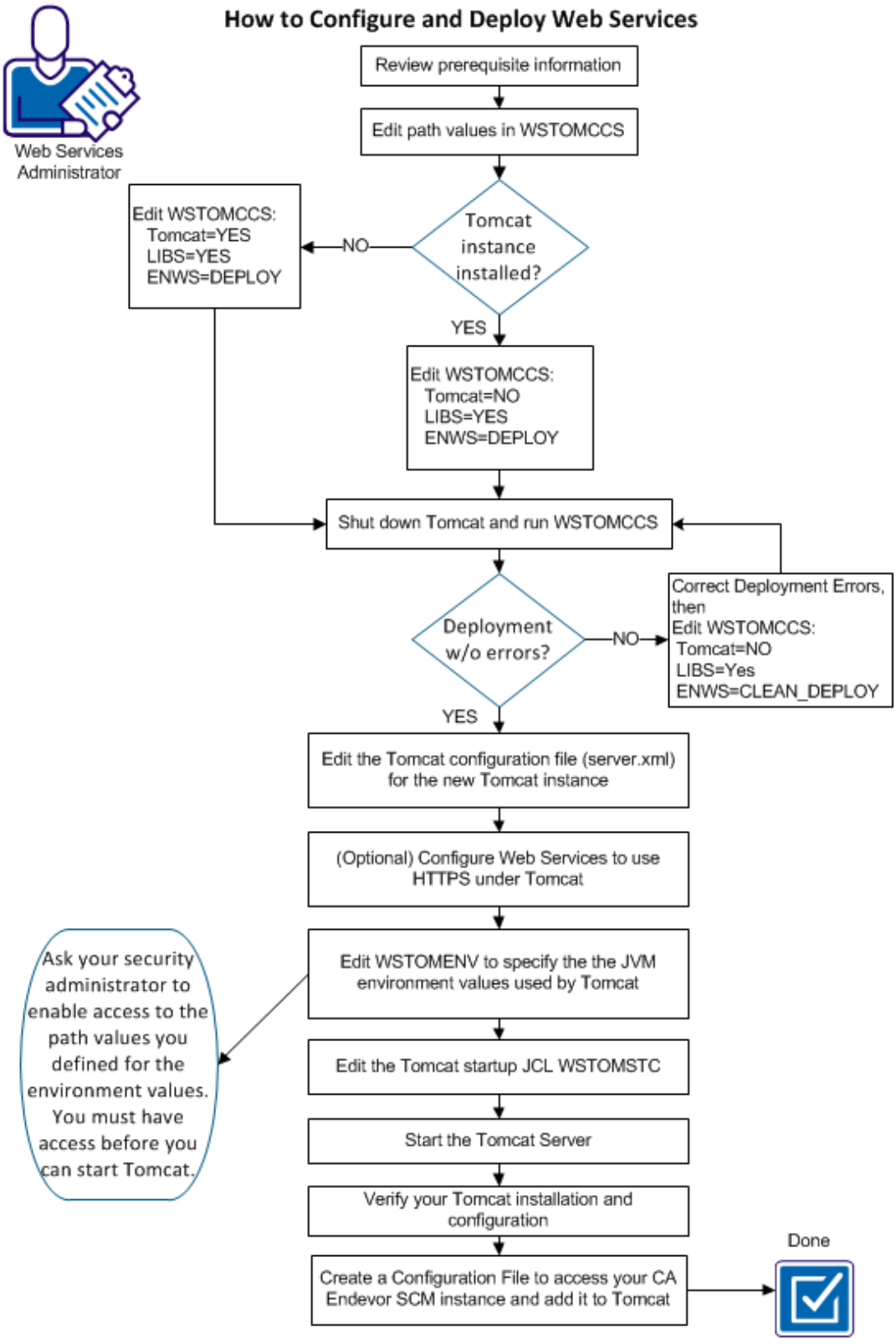
How to Configure and Deploy Web Services

As a web services administrator, you can configure and deploy the Web Services component. This component is a prerequisite for the Eclipse-Based UI. In addition, you can use Web Services to connect your user-written program to the CA Endevor SCM API.

The process of configuring and deploying Web Services on your z/OS system is performed in a UNIX System Services (USS) environment. During this process, you will complete the following objectives:

- Create a dedicated Tomcat instance for Web Services on your Tomcat server.
- Add Runtime libraries (including Axis2 1.6 and Jersey 2.7) to this Tomcat instance.
- Deploy Web Services to the Tomcat instance.

Important: When you want to run Web Services under a new release of Tomcat, we recommend that you create a new Web Services deployment in a new Tomcat directory tree. For more information, see How to Run Web Services Under a New Release of Tomcat (see page 217).


How to Configure and Deploy Web Services
Web Services Administrator
Review prerequisite information
Edit path values in WSTOMCCS
Tomcat instance installed?
NO
Edit WSTOMCCS:
Tomcat=YES
LIBS=YES
ENWS=DEPLOY
YES
Edit WSTOMCCS:
Tomcat=NO
LIBS=YES
ENWS=DEPLOY
Shut down Tomcat and run WSTOMCCS
Deployment w/o errors?
NO
Correct Deployment Errors, then
Edit WSTOMCCS:
Tomcat=NO
LIBS=Yes
ENWS=CLEAN_DEPLOY
YES
Edit the Tomcat configuration file (server.xml) for the new Tomcat instance
(Optional) Configure Web Services to use HTTPS under Tomcat
Edit WSTOMENV to specify the the JVM environment values used by Tomcat
Ask your security administrator to enable access to the path values you defined for the environment values. You must have access before you can start Tomcat.
Edit the Tomcat startup JCL WSTOMSTC
Start the Tomcat Server
Verify your Tomcat installation and configuration
Create a Configuration File to access your CA Endevor SCM instance and add it to Tomcat
Done

To enable Web Services, complete the following steps:

1. Review the prerequisite information (see page 183).
2. Edit the WSTOMCCS member, as described in the member, to add a Tomcat instance for Web Services, add the Web Services files to the new Tomcat instance, and install runtime libraries on your Tomcat server. For more information about WSTOMCCS, see Create or Update a Tomcat directory structure for Web Services (see page 210).
3. Shutdown Tomcat and run the WSTOMCCS job.
4. Edit the Apache Tomcat configuration file. (see page 213) The server.xml file is used to create a Tomcat instance for Web Services.
5. (Optional) Configure Web Services to use HTTPS (see page 215).
6. Edit the JVM environment values used by the Tomcat server.

   Note: You must inform your security administrator of the path values you define for the environment values. Your security administrator must enable access to these paths so that authorized users can use Web Services.
7. Edit the Apache Tomcat Server JCL for Web Services (see page 214). This JCL will enable the started task for Web Services.
8. Start the Apache Tomcat server.
9. Verify your Tomcat installation and configuration. (see page 215) This procedure verifies the Tomcat instance.
10. Create a configuration file, using ENDEVOR.cfg as a template, for the data source (instance of CA Endevor SCM) that Web Services will access. Then deploy the file by saving it to this location: /cai/CADeploy/ESCM/tpv/tomcat/webapps/endevor. For more information, see How Access to a Data Source Works (see page 218) and How to Create a Configuration File (see page 219).

    Note: You can create more than one configuration file, one for each CA Endevor SCM instance at your site.

Note: Completing these steps sets up Web Services to use the Eclipse Plug-in. For more information about client stubs, see User-Written Client Programs (see page 228).

Create a Tomcat Directory Structure for the Web Services Component

Web Services requires a Tomcat instance that can host Web Services. To create and populate the Tomcat instance a script file is provided. This file is **WSTOMCCS** located in CSIQJCL. This job creates an Apache Tomcat instance USS directory structure for Web Services using the CA Common Services for z/OS deployed Tomcat directories and the CA Endevor SCM Web Services install directories as input.

This file can be used to model a script to suit your installation needs. The script has three logical steps that enable you to do the following:

1. Create a Tomcat instance.
2. Add runtime libraries to the Tomcat instance.
3. Add Web Services to the Tomcat instance.

Follow these steps:

1. Modify WSTOMCCS located in CSIQJCL as follows:
   a. Follow the directions in WSTOMCCS to edit the following path values.

      Note: For the USS steps to install and run Web Services these path values are needed. These values will be used in a few different files in this process. The first place these values are needed is WSTOMCCS.

      JAVA_HOME— Specifies the path location of JRE runtime. For example:

      ```
      JAVA_HOME=/sysname/sys/java31bt/v7r0m0/usr/lpp/java/J7.0
      ```

      CCS_TOMCAT_DIR— Specifies the path location of the CA Common Services deployed Tomcat. This path can be part of a file system that is mounted read-only on the system. The default value is:

      ```
      CCS_TOMCAT_DIR=/sysname/cai/CADeploy/CCS/tpv/tomcat.
      ```

      If the product needs a specific release of Tomcat you can change /tomcat to /apache-tomcat-v.r.nn where v.r.nn is the specific release.

      TARGET_TOMCAT_DIR— Specifies the target directory for the Web Service Tomcat instance. This path must be part of a file system that is mounted read-write on the system. The default value is:

      ```
      TARGET_TOMCAT_DIR=/sysname/cai/CADeploy/ESCM/tpv/tomcat
      ```

      ENDEVOR_SOFTWARE_DIR— Specifies the CA Endevor SCM install path. Change the default to the path that the CA Endevor product has been installed to. This path path must end with /software.

      The default value is:

      ```
      ENDEVOR_SOFTWARE_DIR=/sysname/cai/CASoftware/ESCM/software
      ```

b. Follow the directions in WSTOMCCS to do **one** of the following:

Note: If this is your first installation of Web Services, do the first bullet.

- **To *create* a Tomcat instance, *install* runtime libraries on the Tomcat server, and deploy Web Services to the Tomcat instance—** Set the following values:
  - Set the value for TOMCAT to YES.
  - Set the value for LIBS to YES.
  - Set the value of ENWS to DEPLOY.
- **To *update* a Tomcat instance, *install* runtime libraries on the Tomcat server, and deploy Web Services to the Tomcat instance—** Set the following values:
  - Set the value for TOMCAT to NO.
  - Set the value for LIBS to YES.
  - Set the value of ENWS to DEPLOY.
- **To *update* a Tomcat instance, *keep* the runtime libraries that are currently added on your Tomcat server, and redeploy existing Web Services in the Tomcat instance—** Set the following values:
  - Set the value for TOMCAT to NO.
  - Set the value for LIBS to NO.
  - Set the value of ENWS to CLEAN DEPLOY.

  Choose this option when updating a previously failed deployment.

Note: When redeploying an existing deployment, the script removes the Web Services directory trees from the Tomcat instance. Then adds the new directory trees to the target Tomcat instance.

Important— If you have an existing Web Services deployment, all customized files should be backed up before proceeding with this job. That would be any cfg files in your Endevor directory or any other files you have placed in the Endevor directory tree.

Important: When you want to run Web Services under a new release of Tomcat, we recommend that you point to a new Tomcat directory path. After the script has run successfully, update the server.xml file in the new Tomcat tree with the port that you use for Web Services.

2. Run the job.

This job creates all of the directories and files needed to configure and start your Tomcat Server and copies the directory to the instance of Tomcat.

Edit the Tomcat Configuration File

To enable this instance of Tomcat to communicate with Web Services, you must edit the Tomcat configuration file.

Edit the **server.xml** file located here: /cai/CADeploy/ESCM/tpv/tomcat/conf. Change the server, connector, and redirect port numbers as necessary for your site. Make sure that you have a unique TCP/IP port number for this instance of the Tomcat server to use for communications. Each program on a LPAR using TCP/IP for communications must have a unique port number.

The server.xml file is ASCII encoded and it needs to stay that way. One method to edit the server.xml file is to use ISPF EDIT. This method is described next.

Follow these steps:

1. Enter the following command on your ISPF panel: TSO ISH

   The UNIX System Services ISPF Shell panel opens.

2. Enter the following pathname where server.xml is located and then press Enter: /cai/CADeploy/ESCM/tpv/tomcat/conf/

   A directory list opens.

3. Enter an E next to server.xml and press Enter.

   The server.xml file opens.

4. Make your changes to the file and exit the file.

   The file is updated for your site and remains an ASCII encoded file.

Note: If you require an HTTPS implementation of Tomcat, see to How to Configure Tomcat as an HTTPS server (see page 215).

Edit JVM Environment Values Used by Tomcat Server

To establish environment variables for the Apache Tomcat Server, you must edit member **WSTOMENV** in CSIQJCL to specify the values appropriate for your site.

Update the following parameters in member WSTOMENV in CSIQJCL. See the member for detailed instructions on how to update these variables:

- **INSTALL_HOME—** Specifies the location where Tomcat has been installed on the system. This should be based on the value of CCS_TOMCAT_DIR.
- **PRODUCT_HOME—** Specifies the location where Web Services will be implemented. This should be based on the value of TARGET_TOMCAT_DIR.
- **JAVA_HOME—** Specifies the location of the JAVA Runtime Directories used for this instance of Tomcat.

Important: Be sure to inform your CA Endevor SCM administrator and security administrator that the user ID used to start the Tomcat server must have read access to the path defined in INSTALL_HOME and write access to the path defined in PRODUCT_HOME in WSTOMENV.

Start the Apache Tomcat Server

To enable access to Web Services, the JCL to start the Apache Tomcat server must be customized.

Follow these steps:

1. Edit the JCL in member **WSTOMSTC** in CSIQJCL as described in the member:
   - Add a jobcard, or remove the PEND and EXEC statements.

   Note: The STEPLIB value must point to the installed data set CSIQPLD or Web Services will not function. WSTOMENV must be reviewed and updated before trying to start Web Services.

   The WSTOMSTC member is customized.
2. Copy WSTOMSTC JCL from the sample library (CSIQJCL) to a system procedure library so it can be executed as a started task after it has been tested.

   Note: The user ID used to start the Tomcat server must have read access to the path defined in INSTALL_HOME and write access to the path defined in PRODUCT_HOME in WSTOMENV.
3. Work with your systems administrator to make sure that WSTOMSTC task that will be started after an IPL.

   The JCL is customized so that the Apache Tomcat server will start in its own address space.

Verify Your Apache Tomcat Installation and Configuration

Use this procedure to confirm that you have successfully completed the Apache Tomcat installation and configuration procedures. If at any point you do not see the expected result, confirm that you have completed the Web Services configuration steps as documented. If you cannot identify the issue, contact CA Technical Support.

To verify your Apache Tomcat installation and configuration

Open a web browser, enter the appropriate URL listed below, and press Enter

- SOAP Web Services:

  `http://<host>:CA Portal/EndevorService/services/EndevorService?wsdl`

  The Web Services Description Language (WSDL) of CA Endevor SCM SOAP-based Web Services is displayed.

- RESTful API Web Services:

  `http://<host>:CA Portal/EndevorService/rest/application.wadl`

  The Web Application Description Language (WADL) of CA Endevor SCM RESTful API is displayed.

How to Configure Tomcat as HTTPS

You can optionally use HTTPS instead of HTTP for user access. This option lets you specify a user name and password to minimize concerns about the data being exposed in clear text on the network.

To configure Web Services to use HTTPS, follow these steps:

1. Complete the following steps to generate a keystore:
   a. In OMVS, enter the following command:

      `$JAVA_HOME/bin/keytool –genkey -alias tomcat -keyalg RSA`

      A prompt appears.

   b. Specify a password, press Enter, and answer the questions.

      - We recommend using the host name that Tomcat runs on for the CN value so that when you are prompted to accept the certificate, it is clear which server it is coming from.
      - Optionally, specify a different location for the default keystore by replacing */path/to/my/keystore* in the following command:
        $JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA -keystore */path/to/my/keystore*

      A default keystore is created in your home directory with one self-signed certificate inside.

2. Update the Apache Tomcat configuration parameters in the **server.xml** file located in the tomcat_install_dir/conf directory as follows:

   a. Uncomment or replace the SSL connector information to specify site-specific values for the port and keystoreFile parameters.

      Note: Ensure the keystorePass value matches the password specified in Step 1.

      Sample SSL connector data follows:

      ```
      <!-- Define a SSL HTTP/1.1 Connector on port 8443…
      <Connector port="8040" protocol="HTTP/1.1" SSLEnabled="true"
                 maxThreads="150" scheme="https" secure="true"
      clientAuth="false" sslProtocol="TLS"
      keystorePass="Y7ssl"
      keystoreFile="/ca/.keystore"/>
      ```

   b. Edit the redirectPort value in the standard HTTP connector information to match the value specified in the SSL connector data:

      ```
      <Connector port="8080" protocol="HTTP/1.1"
                    connectionTimeout="20000"
                    redirectPort="8040" />
      ```

3. Add the following lines before </web-app> at the end of the **web.xml** file located in tomcat_install_dir/conf:

   ```
   <security-constraint>
       <web-resource-collection>
           <web-resource-name>Tomcat</web-resource-name>
           <url-pattern>*.html</url-pattern>
       </web-resource-collection>
       <user-data-constraint>
           <transport-guarantee>CONFIDENTIAL</transport-guarantee>
       </user-data-constraint>
   </security-constraint>
   ```

After you start the Apache Tomcat server, you will be prompted to indicate whether you trust the defined certificate. Click Yes to import it to your trusted certificates.

Note: For more information about trusted certificates, see Apache Tomcat 6.0 on the Web.

How to Update an Existing Deployment of Web Services

As a web services administrator, you can update an existing deployment of the Web Services component. For example, when you apply Web Services maintenance to the installed Endevor directories you must also deploy the maintenance to the Web Services directories in the Tomcat instance.

To update an existing deployment of the Web Services component, do the following:

1. Back up your configuration files— All customized files should be backed up before proceeding with this job. That would be any cfg files in your endevor directory or any other files you have placed in the Endevor directory tree.
2. Edit the WSTOMCCS member as described in the member to update the Web Services files by setting the following values in WSTOMCCS:
   - Set the value for TOMCAT to NO.
   - Set the value for LIBS to NO.
   - Set the value of ENWS to UPDATE.

   When updating an existing deployment, the script removes the Web Services directory trees from the Tomcat instance. Then adds the new directory trees to the target Tomcat instance.
3. Shutdown the Tomcat server, run the WSTOMCCS job.
4. Copy the saved configuration files into the refreshed directory tree.
5. Restart the Tomcat server.

How to Run Web Services Under New Releases of Tomcat

When you want to run Web Services under a new release of Tomcat, we recommend that you create a new Web Services deployment in a new Tomcat directory tree. To do this,

1. Save copies of your configuration files.
2. Follow the instructions in How to Configure and Deploy Web Services (see page 208) and Create or Update a Tomcat Directory Structure for the Web Services Component (see page 210), specifying new TARGET_TOMCAT_DIR and ENDEVOR_SOFTWARE_DIR paths, and setting the values in WSTOMCCS as follows.
   - Set the value for TOMCAT to YES.
   - Set the value for LIBS to YES.
   - Set the value of ENWS to DEPLOY.

3. Shutdown the Tomcat server, run the WSTOMCCS job.
4. Copy the saved configuration files into the refreshed directory tree.
5. Restart the Tomcat server.

How Access to a Data Source Works

Web Services is delivered as a web archive file (EndevorService.war). Tomcat automatically or manually deploys this .war file. The EndevorService component (also known as a web service end point) has a lifecycle in that it can be started and ended. During the initialization phase, the EndevorService component scans the webapps/endevor folder and identifies and validates the configuration files (*.cfg) that it finds. If a validation failure occurs, the configuration file is discarded. To add, delete, or modify a .cfg file, the EndevorService.war component must be stopped and restarted.

The client application or the Eclipse-Based UI connection wizard can access a list of validated data sources. The list includes the description, status, and message extracted from the configuration file.

A client can have access to more than one validated data source at the same time.

- There can be more than one valid configuration file in the webapps/endevor folder. For example, A1.cfg, A2.cfg, B1.cfg, and B2.cfg could access data sources named: A1, A2, B1, and B2.
- Different configuration files could specify the same started task on the same or on a different CAICCI host. The name of the .cfg file must match the name of the data source specified by the Name value in the .cfg file. However, the name of the data source does not need to be the same as the started task JobName value specified in the .cfg file.

  For example, in the following table, each *.cfg file accesses a data source using the started task on the host computer. All three of these data sources could be available at the same time.

*.cfg	Data Source Name	Started Task Procedure	Host Name
A1.cfg	A1	ENWSTST7	usilnj31
A2.cfg	A2	ENWSTST7	usilnj11
B1.cfg	B1	ENWSTST7	usilcnj31

How to Create a Configuration File

To define a data source (an CA Endevor SCM instance) to Web Services, you must create a data source configuration (.cfg) file. This file includes configuration and connection information about this CA Endevor SCM instance. More than one configuration file can be defined and in use at the same time. The Web Services initialization process scans all .cfg files located in the webapps/endevor folder. Only valid .cfg files are activated.

Follow these steps:

1. Use the supplied .cfg file (ENDEVOR.cfg) as a template or create your own configuration file manually in an editor. This file is coded as an XML document. If the file contains errors, Web Services will not activate it. For more information, see the V17 Data Source Configuration File Template (see page 220).

   Important: The .cfg file is ASCII coded. For more information, see Edit an ASCII Encoded File (see page 219).

   If you are using the supplied ENDEVOR.cfg file template, you must modify the HostName and JobName parameters. Also, you should edit the Comments parameter specific for your installation. Review the other parameters and, if needed, change them to your preferences. For more information, see ENDEVOR.cfg Parameters. All other parameters can be left as defined.

2. Rename the .cfg file to match the value of the Name= parameter coded in the configuration file. The client web service application uses this name to refer to the data source repository.

3. Deploy the file to the following location on your web server:
   /cai/CADeploy/ESCM/tpv/tomcat/webapps/endevor

Edit an ASCII Encoded File

The ENDEVOR.cfg file is an ASCII coded file and it needs to stay that way. One method to edit an ASCII file is to use ISPFEDIT. To edit an ASCII file using ISPFEDIT, do the following:

1. From the ISPF Primary Options Menu access the Utilities menu (Option 3)

   The Utilities Selection panel appears

2. On the Utilities Selection panel access option 17 (Udlist)

   The z/OS UNIX Directory List Utility opens.

3. Enter the pathname where the file you want to edit is located and then press Enter. For example, as delivered, the ENDEVOR.cfg is located here:
   /cai/CADeploy/ESCM/tomcat/webapps/endevor/

   A directory list opens.

4. Enter EA next to the file you want to edit and press Enter.

   An edit entry dialog appears.

5. Select the ASCII data option, and any other options necessary, press enter

   The file opens for edit

6. Make your changes to the file and exit the file.

   Your file is edited and remains an ASCII file.

V17 Data Source Configuration File Template

The delivered data source configuration (.cfg) file is named ENDEVOR.cfg. This file is a template and is ASCII coded. To use this file, you must edit and rename it as appropriate for your site. Code the values of the XML attributes in the copy of the template and save it to the <mount-point>/ESCM/tpv/tomcat/webapps/endevor directory. The template is as follows:

```
<?xml version="1.0"?>
<DataSource xmlns="http://scm.ca.com/endevor/CFG"
            xmlns:cfg="http://scm.ca.endevor"
            Name="ENDEVOR"
            Status="Available"
            Message="Welcome to ENDEVOR"
            Description="Endevor demo">
  <STC HostName="CCIHOST"
       JobName="CCISTC"
       ProgramName="BC1PAPI0"
       Comments="ENDEVOR STC"
       PoolInitSize="0"
       PoolIncrSize="1"
       PoolMaxSize="0"
       PoolreapTime="180">
       UnusedTimeout=”600”
       AgeTimeout=”1200”>
    <wsParameters Lang="EN"
                  TimeZone="GMT-6.0"
                  CodePage="cp1040"
                  Encoding="cp1040"
                  CharacterSet="ISO8859-1"
                  LFSEncoding="ISO8859-1"
                  ContentType="Text"
                  Trace="1"
                  Traced="USERID"/>
  </STC>
</DataSource
```

ENDEVOR.cfg Parameters

The delivered data source configuration file, ENDEVOR.cfg, file contains the recommended parameters for running web services. The following list describes each of these parameters, as well as additional, optional and reserved parameters:

- **DataSource parameters** – Specify the name of the data source and information about the data source. The parameters are as follows:

  Name

  Specifies a 1- to 20-character alphanumeric name that identifies the CA Endevor SCM data source that this configuration file applies to.

  Note: To be a valid configuration file, the name of the .cfg file must match the Name= value specified within the file. To connect a user-written client to a data source, the DataSource parameter specified in the client request is matched to the name of the configuration file.

  Status

  Specifies whether the service is active. Valid values are 1- to 15-character text strings. Recommended values are Available or Unavailable. This text appears where the data source is listed as a valid data source by the client application.

  Message

  Specifies any 1- to 100-character message you want for this data source, such as "Welcome to CA Endevor SCM Web Services. Service is open." This text appears where the data source is listed as a valid data source by the client application.

  Description

  Specifies any 1- to 100-character description you want for this data source, such as "CA Endevor SCM Web Service" This text appears where the data source is listed as a valid data source by the client application.

- **STC parameters** – Specify the CCI host where CA Endevor SCM is running, the CCI PROCNAME defined in the started task procedure to be spawned, and the program name for CA Endevor SCM on the specified host z/OS mainframe.

  Hostname

  Specifies the one- to eight-character CAICCI hostname associated with the data source. Use the value of SYSID in the CCIPARM file, which is the SYSID of the system where CA Endevor SCM is installed.

  Jobname

  Specifies the one- to eight-character CAICCI host application or process name used to start the CA Endevor SCM API server. Use the CA Endevor SCM mainframe started task procedure specified in the WSEWSCCI file. The jobname supplied as an example is: WSEWSSTC

ProgramName

Specifies the one- to eight-character name of the program started by Jobname, which is BC1PAPI0. This is the only valid value.

UserId (Optional)

Specifies the one- to eight-character user ID used to spawn STCs for the STC pool. Any ID used here must be defined following the instruction in “How to Enable Security Access to Web Services”. This value overrides the user ID used to spawn STCs. The default user ID that is used to start the Web Services with a password of NOTREQED.

Password (Optional)

Specifies a one- to eight- character string. The value is either the password for the ID supplied in UserID or NOTREQED if that ID does not have a password. This value or EncryptedPassword must be coded if UserId is coded.

EncryptedPassword (Optional)

Specifies a 64-bit encoded string of the Password value. For a Password of NOTREQED this value is Tk9UUkVRRUQ=. This value or Password must be coded if UserId is coded.

Comments

Specifies a 1- to 100-character string that helps identify the data source to users.

PoolInitSize

Specifies the number of initial (permanent) STC in the STC pool. The permanent STCs are created when Web Services is initiated. Each STC in the pool can transmit API requests from different clients to CA Endevor SCM on the mainframe.

Valid values are: 0 through 20

Default: 0

Note: The sum of PoolInitSize and PoolIncrSize cannot exceed the value of PoolMaxSize.

PoolIncrSize

Specifies the number of incremental STCs that the pool maintenance thread can create in addition to the permanent pool when a shortage of STC is detected. If PoolInitSize is set to 0, all STC connections are allocated from the incremental Pool. After the workload decreases, the thread releases the extra STC connections from the incremental pool. STCs will be started one at a time as needed.

Valid values are: 0 through 5

Default: 1

Note: The sum of PoolInitSize and PoolIncrSize cannot exceed the value of PoolMaxSize.

PoolMaxSize

Specifies the maximum number of live permanent and incremental STCs in the pool. The STC pool maintenance thread maintains up to this number of STCs in the pool. If the maximum numbers of STCs are fully utilized, incoming web service calls are rejected.

Valid values are: 0 through 40

Default: 0

PoolreapTime

Specifies the interval, in seconds, between runs of the STC pool maintenance monitoring thread. For example, if this value is set to 180 (the default), the monitor thread runs every 180 seconds. Consider the following criteria when setting the PoolreapTime value:

- This value also affects performance. Smaller intervals mean that the STC monitor thread runs more often and degrades performance.
- To disable the pool maintenance thread, we recommend setting PoolreapTime to 0. In this case, Unused Timeout and Aged Timeout are ignored. Alternatively, you can disable pool maintenance by setting both Unused Timeout and Aged Timeout to 0. In this case, the pool maintenance thread runs, but only physical connections which timeout due to non-zero timeout values are discarded.

Valid values, in seconds, are: 0 through 600

Default: 180

Unused Timeout

Specifies how long an STC connection can remain idle, before the monitor thread discards the connection. The accuracy of this timeout and performance are affected by the PoolreapTime value.

Set the Unused Timeout value higher than the PoolreapTime value for optimal performance. For example, if the unused timeout value is set to 120 and the STC monitoring thread is enabled with a PoolreapTime of 60, any unused STC remains active for at most 60 seconds before being shut down.

Valid values, in seconds, are: 0 through 3600

Default: 0

Aged Timeout

Specifies how long an STC connection can remain active, before the monitor thread discards the connection. The accuracy of this timeout and performance are affected by the PoolreapTime value.

Set the Aged Timeout value higher than the PoolreapTime value for optimal performance. For example, if the Aged Timeout value is set to 1200 and the PoolreapTime value is 60, any STC remains in existence for at most 1140 seconds (19 minutes) before being shut down.

Setting Aged Timeout to 0 allows active STCs to remain in the pool indefinitely, unless shut down is based on other parameter values.

Valid values, in seconds, are: 0 through 1200

Default: 1200

Connection Timeout

This parameter is for future use. The only valid value is 0.

Default: 0

ConnectionMaxWS

This parameter is for future use. The only valid value is 0.

Default: 0

ConnectionMaxKBytes

This parameter is for future use. The only valid value is 0.

Default: 0

ResponseTimeout

This parameter is for future use. The only valid value is 0.

Default: 0

KeepAlive

This parameter is for future use. The only valid value is 0.

Default: 0

- **wsParameters parameters** – Specify default values that are appended to the submit call. These parameters detail the handling of the data transported on the Internet from or to CA Endevor SCM.

  Important: These values are superseded if values are passed with the submitSCL() call.

  Lang

  Specifies a two-character language code for the CodePage.

  Default: EN.

  TimeZone

  Specifies the time zone where the data source is located, expressed as a positive or negative offset from GMT+0, for example GMT-6.0. Web Services uses this value to determine the correct time on the z/OS mainframe where the data source resides, for use in messages.

  CodePage

  Specifies the code page, defined in the mainframe operating system that CA Endevor SCM is using. The code page is a variation of the Extended Binary Coded Decimal Interchange Code (EBCDIC) that defines how text files are coded on the mainframe.

  The CodePage and CharacterSet parameters enable Web Services to translate the plain text information exchanged between the client computer and the API. Plain text information includes SCL commands sent to the API and reports and CSV files extracted from the API.

  For information about supported encodings, see:
  http://download.oracle.com/javase/1.5.0/docs/guide/intl/encoding.doc.html

  Default: cp01140

  CharacterSet

  Specifies the character set used by the client application, for example latin1 or iso8859-1. The CharacterSet value specifies how Web Services is to convert the mainframe code page to and from the alphabet used by the client application. This value is used for conversion of plain text information only. For more information, see the CodePage parameter defined in this topic.

LFS.Encoding

Specifies the encoding scheme for CA Endevor SCM elements stored in the client local file system. This value specifies the name of the character set that the Web Services Java Virtual Machine uses for text character conversion. This value applies to the exchange of files between the client application and the API. In CA Endevor SCM, these files are associated with an element Type defined with DATA FORMAT=Text.

- When Adding or Updating a text file from a local file system to CA Endevor SCM, the data of the text file is first converted from the LFS.Encoding scheme into the Unicode character set, and theninthe target character set defined by the Encoding parameter.
- When Retrieving a text file, the data is first converted from the CA Endevor SCM Encoding scheme into the Unicode character set, then into the target character set specified by the LFS.Encoding parameter.

Default: ISO-8859-1

Important! When you Add or Update an element to a file Type defined in CA Endevor SCM as DATA FORMAT=Text, Web Services attempts to translate this file from the code page on the local workstation to the mainframe code page using the character sets defined in the LFS.Encoding and CodePage parameters. All characters in the local code page may not have an appropriate mapping in the host code page. The same consideration applies when retrieving a text element. For files that do not require character set conversion, the Type definition must specify DATA FORMAT=Binary.

Encoding

Specifies the encoding scheme for CA Endevor SCM element. This value is usually the same value as the CodePage.

ContentType

Specifies how CA Endevor SCM reports and queries are exchanged with the Web Services client computer. The default is Text.

Trace

Specifies what to trace. Valid values include a trace level or string value. The default is 0.

Traced

Specifies the user name to be traced. Valid values include:

user-name – Specifies a one- to eight-character user name.

ALL – Specifies all users.

How to View the Content of a Data Source Configuration

You can use one of the following methods to view the content of a data source configuration file.

- The getConfiguration service lets you request the parameters of a data source configuration file while Web Services is live. This service returns an XML document that shows the configuration parameters in the memory image of the configuration. Regardless of the content of the XML document, you can always use the following method to view the disk image.
- To view the disk image, open the configuration file in the /endevor folder under the z/OS USS environment that hosts Web Services.

How to Enable STC Pooling

To enable STC Pooling the PoolMaxSize must be set to a value greater than zero. This allows STCs to remain active so Web Services can pass multiple requests without spawning additional STCs. During the activation of Web Services when all configuration (.cfg) files located in webapps/endevor directory are fetched, parsed, and loaded into memory, the value in PoolMaxSize is used to initialize control information used in pooling. If a value greater than zero is coded for PoolInitSize, that number of STCs are spawned as part of the startup. By default the userID that started Tomcat will be used to spawn STCs with a password of NOTREQED.

Note: If a .cfg file fails the parsing validation it will not be loaded into memory and will not be available for use.

Note: All CA Endevor SCM Web Services requests are switched to the ID of the requester for the duration of the action.

The delivered ENDEVOR.cfg configuration file template does not have STC pooling turned on. To enable this feature, it is reasonable to start with the following values in the cfg file:

- PoolInitSize=2,
- PoolIncrSize=1,
- PoolMaxSize=5.

Make the changes to the .cfg file and restart Tomcat. Monitor Tomcat and your users to identify which values require adjustment.

Important: Before turning on STC pooling or making any changes to the supplied values, make sure that you understand how the parameters affect processing or consult CA Support.

How the STC Pool is Managed

A Tomcat internal (monitor) thread manages the STC pool. The PoolreapTime parameter sets the interval at which the monitor thread runs, for example, every 60 seconds. Each time it runs, the monitor thread checks the status of the STCs in the pool.

The STC pool is a group of STC connections to the backend z/OS region executing the CA Endevor SCM API. The following STC parameters determine the pool size:

- **PoolinitSize—** Specifies the number of permanent STC in the STC pool. The permanent STCs are created when Web Services is initiated. When all the permanent STCs are in use, incremental STCs can be created.
- **PoolIncrSize—** Specifies the number of incremental STCs that the monitor thread can create in addition to the initial pool when a shortage of STC is detected. If PoolInitSize is set to 0, all STC connections are allocated from the incremental Pool. To anticipate a new workload, the monitor thread creates additional STC connections. The PoolIncrSize parameter specifies the number of additional connections that a thread can create. After the workload decreases, the thread releases the extra STC connections from the incremental pool.
- **PoolMaxSize—** Specifies the maximum number of live permanent and incremental STCs in the pool.

Using Web Services

Web Services supports two methods for accessing the CA Endevor SCM API. The SOAP method is built using the Axis2 framework. This method supports most of the CA Endevor SCM actions. The RESTful method is simpler to use, but only supports Package actions and some list actions.

CA Endevor SCM SOAP Web Services

The SOAP client design model requires a client side stub. The web developer must create a client stub from the CA Endevor SCM Web Services WSDL (Web Services Description Language) file. WSDL is a service description protocol. The WSDL file is a dynamic XML file that serves as the interface between a client program and the web service. The Web Service WSDL file describes what CA Endevor SCM operations are available to the client program and the SOAP protocol bindings and message formats required to enable the client to interact with Web Services.

The client stub is responsible for conversion of parameters used in a function call and deconversion of the results passed from the server after execution of the function. The client stub interacts with the web service stub.

The program developer writes the client program to collect the following information:

- A login user ID and password for access to a specified CA Endevor SCM repository.
- An SCL statement with optional wsParameters.
- If this is a submitSCL request, then parameters for a SOAP Attachment to transport data to and from Web Services are also required.

Depending on the programing language of the client program, details for generating the client stub can differ, but conceptually the process is very similar. A stub generator is used to produce a stub for a particular programming language. Then the stub is included in the source code.

- For client programs written in C, the stub, in C code, is compiled and the generated header files are required to code the application program. The application program must have the #include structure.h in C++ include wobject.h>
- For client programs written in Java, the stub is a collection of Java sources grouped in a package. The sources are compiled by the javac compiler into a JAR file. This JAR file is required by the sample code. The program code must include an import statement that refers to the generated packages (this allows the java IDE to make the Loginproperties object available to the application program). The client program can instantiate the stub to invoke the web service. For a client to access the stub, the stub class should be present in the classpath of the client application.

  A Java client program instantiates the following objects and populates them with the correct values for a specific request.

  - LoginProperties
  - SCL
  - Attachment

  The client stub created from the Web Services WSDL supports the instantiation of these objects. The data in these objects is required by Web Services. The client program populates the objects with the required parameters.

How to Create a Client Stub

The SOAP client design model requires a client stub to enable the client program to access Web Services. The client application developer must create a client stub, using the Web Services WSDL file. Then the client-side stub must be incorporated as an import package into the client program. The stub manages, sends, and receives SOAP messages to communicate with the web service.

- You can create the client stub manually or using a utility. After Web Services is installed, the WSDL file can be accessed using a web browser and saved as an xml file. There are third-party tools, for example WSDL2Java (part of the Axis2 distributions) that can be used to generate a client side stub from the WSDL.

  The WSDL file for Web Services is located at:

  ```
  http://<servername>:<portnumber>/EndevorService/services/EndevorService?wsdl
  ```

  Replace the <servername>:<portnumber> specifications with the name and port number of the web server where your installation of Web Services resides.

- After the stub is written or generated, the client stub has to be compiled and the client application's logic implemented above the stub. The client program must instantiate the stub to invoke the web service. For a client to access the stub, the stub class needs to be present in the classpath of the client application.

Java Client Program Objects and Parameters

A Java client program populates the objects with the following required parameters:

- **LoginProperties object** – Contains the following parameters:

  DataSource

  Specifies the data source repository (CA Endevor SCM) that the client wants to access. This value must match the name of a valid configuration file accessible by Web Services. For more information, see How to Create a Configuration File (see page 219) and How Access to the Data Source Works (see page 218).

  UserID

  Specifies the user ID for access to the CA Endevor SCM API.

  Password

  Specifies the password for access to the CA Endevor SCM API.

- **SCL object**–Contains the following parameters:

  Statement

  Specifies the SCL statement entered by the user of the client application.

  For any get*Object* service request, a Comma Separated Value utility List statement is required. For more information about List statements, see "Using the Comma Separated Value (CSV) Utility" in the *Utility Guide*.

wsParameters

(Optional.) Specifies parameters and values to override wsParameters specified in the configuration file. These parameters are Lang, TimeZone, CodePage, LSF.Encoding, Encoding, CharacterSet, ContentType, Trace, and Tracer. For more information, see Configuration File Parameters.

Category

Specifies the type of SCL request. *All* the SCL specified in a request must be for the same type. Valid values follow:

A – Specifies environment administration actions.

C – Specifies Comma Separated Value (CSV) utility actions. Required for any get*Object* service request.

E – Specifies element actions.

P – Specifies package actions.

S – Specifies package ship actions.

X – Specifies batch ACM Query utility actions.

L – Specifies Add, Update, or Retrieve requests that required local file system (LSF) support. The SCL statements for these actions require the To | From Attachment clause to specify the location on the local client where the data will be transferred to or from.

Note: Use Category E for Add, Update, or Retrieve requests that are *not* LFS requests (with the Attachment clause) and that use either To | From DSName or To | From Path HFSFile.

Timezone

Specifies the timezone where the client application is located. Web Services uses this value to determine the correct time, on the client side, for use in messages.

- **Attachments object** – Contains the following parameters:

DD

Specifies the data set on the client side that contains the data being exchanged.

Note: For the base64Binary type, the DATA are coded with the 64Binary translation and manipulated by a data handler or without any translation and appended as a MTOM attachment. The client code is expected to fill the DD property using a data handler (in the case of the 64binary translation) or as MTOM attachment to take advantage of the streaming.

DDName

Specifies the attachment name that references what is being exchanged.

contentType

Specifies the data type being exchanged. Specifies how the data transported by the Attachment objects of the client call are coded. This value informs Web Services how to format the data part of the attachment. ContentType is a standard of the Multipurpose Internet Mail Extensions (MIME) message protocol used by HTTP for complex messages, including messages with file attachments.

Example: Client Program in Java for the SubmitSCL Service

A Java client program invokes a stub (generated from the WSDL) and runtime (AXIS2/Java). The following sample program for the submitSCL service instantiates the loginProperties object, the SCL object, and the Attachments object. The client program populates the objects with the values required for a specific client request.

The request includes the login properties for access to a specified CA Endevor SCM repository. The request also includes the SCL statement and optional parameters, as well as a SOAP Attachment, used to submit an SCL statement and data to Web Services.

```
public void consume() {
    setException(null);
    int attachmentCount = 0;
    /*
     * instantiate the loginProperties object
     */
    LoginProperties loginProperties = new LoginProperties();
    loginProperties.setDataSource(this.getDataSource());
    loginProperties.setUserId(this.getUserId());
    loginProperties.setPassword(this.getPassword());

    /*
     * instantiate the SCL object
     */
    SCL scl = new SCL();
    scl.setStatement(this.getStatement());
    scl.setWsParameters(this.getWsParameters());

    /*
     * instantiate the Attachments object
     */
    if (inputAttachments != null)
        for (int attachmentNumber = 0; attachmentNumber < inputAttachments.length; attachmentNumber++) {
            Attachments attachment = new Attachments();
            attachment.setDDName(inputAttachments[attachmentNumber]
                    .getDDName());
            attachment.setContentType(inputAttachments[attachmentNumber]
                    .getContentType());
            FileDataSource dataSource = new FileDataSource(
                    inputAttachments[attachmentNumber].getInputFileName());
            DataHandler dataHandler = new DataHandler(dataSource);
            attachment.setDD(dataHandler);
            scl.addAttachments(attachment);
        }

    SubmitSCL submitSCL = new SubmitSCL();
    submitSCL.setLoginProperties(loginProperties);
    submitSCL.setScl(scl);
```

Valid Operations for User-Written Client Programs

User-written programs use Web Services to send Software Control Language (SCL) statements (with some restrictions) to the CA Endevor SCM API. In addition, information can be retrieved from Master Control File, Package File, and repository objects. For more information, see SubmitSCL Operation (see page 234) and Get<Object> Operation (see page 236).

SubmitSCL Operation

Web Services supports the submitSCL operation. Client requests can communicate with the CA Endevor SCM API using the CA Endevor SCM Software Control Language (SCL). With some restrictions, any valid SCL statement can be sent to the CA Endevor SCM API and the results will be returned to the client, providing full access to the repository.

Important: The SCL syntax requires one or more blank spaces before the period that ends the SCL statement.

Valid SCL statements for the submitSCL operation for Web Services are subject to certain restrictions.

- The following actions are *not* valid:
  - &&ACTION
  - Archive element
  - Copy element
  - List member From Archive
  - Print member From Archive
  - Restore element
  - Transfer To Archive
  - Transfer From Archive
  - Archive Package
  - Export Package
- The parameter DDNAME is generally *not* supported for Web Services. However, DDNAME C1PRINT *is* valid on List and Print actions.

  Note: The DSName *dataset-name* parameter *is* supported.

  Instead of the DDName parameter, the Add, Update, and Retrieve actions use SOAP attachments to transport data between the client and the mainframe. For more information, see To | From Attachment Clause on SCL Statement (see page 235).

Note: No footprint information is stored in files retrieved from CA Endevor SCM. Temporary files written to the Web Services server are purged at the start of each new local file system command.

To | From Attachment Clause on SCL Statement

Every web service call (request) from a client application requires a SOAP Attachment object to submit requests and receive responses. If the request transports data between the client and Web Services, the SCL statement must include a From Attachment or To Attachment clause. This clause specifies the name and location of the file on the client side that is being sent from the client or to the client.

The following SCL requests require the From Attachment clause:

- Element actions Add or Update.
- Upload of package SCL statements.

The following SCL requests require the To Attachment clause:

- Element action Retrieve.
- Fetch of reports.
- Extract of Comma Separated Value (CSV) data.

To | From Attachment Syntax

Instead of the TO | FROM DDNAME clause on the SCL statement for an Add, Update, or Retrieve action, use the following syntax:

```
TO|FROM ATTachment attachment-name PATH mypath LFSFILE myfile
```

To Attachment *attachment-name*

Specifies the name of the SOAP Attachment object that contains the data that is to be sent from CA Endevor SCM to the client.

From Attachment *attachment-name*

Specifies the name of the SOAP Attachment object that contains the data that is to be sent from the client to CA Endevor SCM.

PATH mypath

Specifies the directory on the client side. For the From Attachment clause, Path is where the source file resides. For the To Attachment clause, Path is where the file is to be saved to.

LFSFILE myfile

Specifies the name of the file in the directory on the client side. For the From Attachment clause, Lfsfile is the name of the source file. For the To Attachment clause, Lfsfile is the name that the received file is saved as.

Example: Add Element Statement with Attachment

The following Add Element statement adds element CCIDRPT2 to Stage 1 of the DEV Environment, EABASE System, UTILITY Subsytem, COBOL Type library. The SOAP Attachment object UPLOADD contains a copy of the content of *myfile* located at *myPath.* on the client side. This statement adds the contents of the SOAP Attachment to CA Endevor SCM. The optional Options clause associates the comment "Project myfile' to the element.

```
ADD ELEMENT CCIDRPT2
    FROM ATTACHMENT UPLOADDD PATH myPath LFSFILE myFile
    TO ENV DEV SYS EABASE SUB UTILITY TYPE COBOL STAGE NUMBER 1
    OPTIONS UPD COMMENT 'Project myFile' .
```

Example: Retrieve Element Statement with Attachment

The following Retrieve Element statement retrieves element CCIDRPT5 from Stage 1 of the DEV Environment, EABASE System, UTILITY Subsytem, COBOL Type library. A copy of the data that is element CCIDRPT5 is added to the SOAP Attachment object DNLOADD. Web Services send a SOAP message response back to the client. The SOAP message includes Attachment. The client extracts the data from the Attachment and copies it to *myfile* located at *myPath.* on the client side.

```
RETRIEVE ELEMENT CCIDRPT5
    FROM ENV DEV SYS EABASE SUB UTILITY TYPE COBOL STAGE NUMBER 1
    TO ATTACHMENT DNLOADDD PATH myPath LFSFILE myFile .
```

Get<Object> Operation

Web Services supports various get<Object> services that extract information in the form of an object. Information can be acquired from the Master Control File, Package File, or repository objects. The get<Object> service is based on CSV LIST requests and makes use of a native SOAP facility for creating objects to send and decoding the CSV output that is returned.

The get<Object> services are designed like the submitSCL service. Each get<Object> service retrieves a specific type of information. The <Object> part of the service name indicates the type of information that the service can retrieve. Each get<Object> service requires a corresponding SCL statement that has been appropriately coded to extract the requested information. Each get<Object> service returns an object designed from the information extracted by the Comma Separated Value (CSV) utility as directed by the SCL statement. The extracted information is converted into XML format and returned to the client.

In terms of object-oriented programming, the get<Object> service returns an array, designated as Object[]. Each entry in this array includes one occurrence of the retrieved objects and the object contains the requested information.

To view the property names that can be returned for each get<Object> service, see the WSDL file. For more information about the List statements and the corresponding returned property names and values, see the "Using the Comma Separated Values Utility" chapter in the *Utilities Guide*. The <Object> are built from the CSV representation, their properties map with the CSV layout. The property names match one by one with the CSV columns as described by the WSDL file.

The returned property names and values are the same as those returned by the CSV utility. However, in the case of an object (rather than a CSV request output) the properties are typed as string, integer, or date. The WSDL defines the type of each property. The date and time stamp is the external representation of the CA Endevor SCM stamp coded using the z/OS time zone.

Note: For more information about the Web Services component, see the chapter "Web Services" in the Scenario Guide. This same content can be found in the scenario *How to Enable Web Services,* which is accessible from the Knowledge Based Articles section on the documentation bookshelf.

Get<Object> Syntax for SCL Statements

Each get<Object> service requires a specific type of SCL statement. These SCL statements are the same as those used by the CSV utility. If a service request includes an incorrect SCL statement or an SCL statement that does not correspond to the get<Object>, the request will be rejected. For more information about the options available for each service, see the syntax for the corresponding List statement in the chapter Comma Separated Values in the *Utilities Guide*. For example, for all the optional clauses and parameters that can be coded on the SCL statement for the getElementBasic operation, see The List Element Function in the *Utilities Guide*.

Important: The SCL syntax requires one or more blank spaces before the period that ends the SCL statement.

The following get<Object> services are supported. The minimum required SCL syntax is shown next with each operation.

getApproverGroup

LISt APProver GROup *approver-group-name*

getApproverGroupJunction

LISt APProve GROut JUNction FROm *options*

getConfiguration

LISt CONFIguration

getConfigurations

LISt CONFigurations

getDataSet

LISt DSName FROm *options*

getDestination

LISt DEStination *destination-id*

getElementBasic

LISt ELEment *element-name* FROm *options* DATa BASic

getElementDetail

LISt ELEment *element-name* FROm *options* DATa ALL

getElementSummary

LISt ELEment element-name FROm *options* DATa SUMmary

getEnvironment

LISt ENVironment *environment-name*

getPackage

LISt PACkage *package-id*

getPackageActionSummary

LISt PACkage ACTion FROm PACkage *package-id*

getPackageApproverGroup

LISt PACkage APProver GROup FROm PACkage *package-id*

getPackageSCL

LISt PACkage SCL FROm PACkage *package-id*

getPackageShip

LISt PACkage SHIp FROm PACkage *package-id*

getProcessorGroup

LISt PROcessor GROup *processor-group-name*

getSystem

LISt SYStem *system-name*

getSubSystem

LISt SUBsystem *subsystem-name*

getStage

LISt STAge *stage-id*

getType

LISt TYPe *type-name*

How the Get<Object> Service Works

Web Services supports various get*<Object>* services that extract an array of information. The information is returned as an object-oriented programming object representation of the CSV row extracted by the SCL statement List. To acquire this array, you must code a Java client program that invokes a stub (generated from the WSDL) and runtime (AXIS2/Java).

The logic and coding style of a program for the get<Object> service is similar to the submitSCL() service. The program includes the following:

- The initialization of the loginProperties object and the SCL object, populated with ad hoc information
- The invocation of the web service by its name
- The fetch of the results as an array of specialized objects and attachments containing the API reports

Example: Client Program in Java for the getTypeService

The following sample Java program illustrates the getType Service.

```
public void consume() throws Exception {
    // Thread.sleep(3000);
    setException(null);
    /*
     * instantiate the loginProperties object
     */
    LoginProperties loginProperties = new LoginProperties();
    loginProperties.setDataSource(this.getDataSource());
    loginProperties.setUserId(this.getUserId());
    loginProperties.setPassword(this.getPassword());

    /*
     * instantiate the SCL object
     */
    SCL scl = new SCL();
    scl.setStatement(getStatement());

    scl.setWsParameters(getWsParametersTypeC());
    GetType getType = new GetType();
    getType.setLoginProperties(loginProperties);
    getType.setScl(scl);

    resultType = stub.getType(loginProperties, scl);
    result = resultType.getResult();
    Type type[] = resultType.getType();
    String siteId = type[0].getSiteId();
```

CA Endevor SCM RESTful API

The CA Endevor SCM RESTful web services API (RESTful API) is an extension of the SOAP based Web Services feature. You can build a client program using the RESTful method as an alternative to the SOAP-based method.

The RESTful API provides the Representational State Transfer (REST) architectural style of web services, which uses HTTP 1.1 and Uniform Resource Identifiers (URI). This architectural style provides a simpler, more lightweight API, scalability, improved performance, and easier adoption due to better known distributed/server-client application development model based on the HTTP protocol.

In the RESTful architectural style, date and functionality are considered as resources and are represented and accessed by URIs. These resources can be manipulated by a set of well-defined operations: Create, Read, Update and Delete. In the case of HTTP, these operations are the Request methods GET, PUT, POST, and DELETE.

The RESTful API supports the following Package processing actions and supporting List actions.

Available Package Actions:

- Create/Update
- Cast
- Approve/Deny
- Reset
- Commit
- Delete
- Backout/Backin
- Execute
- Ship
- List Packages

Available Inventory Actions:

- List Repositories
- List Systems
- List Subsystems
- List Elements

RESTful API Client Program Design Considerations

When you design your client program, the following considerations apply.

- **Setting requests and processing responses—** The RESTful API design model is an architectural style of web services that takes advantage of existing and well know concepts of the World Wide Web and the usage of the HTTP protocol. Thus, there are many ways to use the model. The Web Services RESTful API requires the Jersey implementation of the JAX-RS specification. You can use the Jersey framework to develop client applications that are based on the RESTful API. However, you are free to use any applicable method for setting requests and processing responses.
- **Authentication—** The RESTful API uses the HTTP basic access authentication method, which requires a user name and password when creating a request. When creating a HTTP request, the client program uses the Authorization header which is created in the following manner:

  1. The username and password are combined into this string: "username:password".
  2. Then the string is encoded using a Base64 encoding scheme. (Full specifications for the encoding scheme are contained in RFC 1421 and RFC 2045 - MIME (Multipurpose Internet Mail Extensions).)
  3. Then the authorization method and space, for example "Base" , is placed before encoded credentials.

  For example, if the user name is "ndvrusr" and the password is "mfpsswd" then the final header is as follows:

  ```
  Authorization: Basic bmR2cnVzcjptZnBzc3dk
  ```

  If the credentials are not valid the "HTTP 401 Not Authorized" response code is returned.

  Important: Although the username and password are not humanly readable, this method does not encrypt credentials. To secure the communication, use SSL/HTTPS.

- **SSL/HTTPS—** To ensure secure communication and encryption of the messages, configure and use SSL/HTTPS. When working with SSL/HTTPS, ensure the correct protocol and port is used. For more information about setting up HTTPS, see How to Configure Tomcat as HTTPS (see page 215).
- **WADL—** Jersey supports the Web Application Description Language (WADL). WADL is an XML description of the deployed RESTful web application. It contains the model of the resources, structure, supported media types, HTTP methods, and so on. It is similar to the Web Services Description Language (WSDL), which is used for the SOAP-based Web Services model. WADL can also be used for generating a client-side stub, however this feature is not as sophisticated as the stubs generated using the WSDL.

  To list the WADL for the RESTful API use following request:

  ```
  http://<server>:<portnumber>/EndevorService/rest/application.wadl
  ```

RESTful API Outputs and Return Codes

After an action is submitted to CA Endevor SCM, the user will get one of the following returns:

Success:

```
{
  "returnCode" : "####",
  "reasonCode" : "####",
  "data" : []
}
```

Error:

```
{
  "returnCode" : "####",
  "reasonCode" : "####",
  "messages" : [ "Error Message" ]

}
```

The returns use standard HTTP error code syntax. Any additional information is included in the body of the JSON-formatted response. The HTTP response codes corresponds to the CA Endevor SCM return codes as shown in the following table:

CA Endevor SCM Return Code	HTTP Response Code
0000	200 OK
0004	206 Partial Content
0008	409 Conflict
0012	400 Bad Request
0016	500 Internal Server Error

RESTful API Action Request URIs

The RESTful API can process requests for the following types of information:

- Repository
- Package
- Inventory

The RESTful model uses HTTP 1.1 and Uniform Resource Identifiers (URI). In the RESTful architectural style, data and functionality are considered resources, which are represented and accessed by URIs. These resources can be manipulated by a set of well-defined operations: Create, Read, Update, and Delete. To perform these operations on the resources, your client program uses the HTTP Request methods GET, PUT, POST, and DELETE. Which method is used depends on the action you want to perform.

The URI (Uniform Resource Identifier) for each request includes the URL (Uniform Resource Locator) to define the resource location. Some URIs also include the URN (Uniform Resource Name), depending on the action. Each URL includes the same base URL, the URL specific to a particular action, and may include action parameters. For more information about the action URLs and the action parameters for a particular action, see the descriptions for each action.

Details follow about the base URL, rules for building query strings for action parameters, and name-masking of certain variables.

- **Base URL—** All RESTful actions have the same base URL, which is represented by the protocol, the host or server name, the appropriate port number, the EndevorService identifier, followed by the resource identifier (rest) for the RESTful Web Service root.

  ```
  <base URL> = <protocol>://<host>:<portnumber>/EndevorService/rest
  ```

  protocol

  Specifies whether http or https is enabled for your client program. Valid values are http or https.

  host

  Specifies the host /server name, or the domain name where the service is deployed and running.

  Portnumber

  Specifies the port number defined on the web server.

  Important: All URLs are case sensitive!

  Note: In the action syntax descriptions, <base URL> is specified to simplify the descriptions.

- **Action Parameters—** You can specify parameters on some of the action requests to influence action execution. For information about the parameters for a particular action, see the description for that action.

  To use the parameters, follow these basic rules for URL query string building:

  - The query string is constructed as a series of parameter-value pairs that are located after the URI using the '?' character as a separator. ('?' is not part of the query string.)
  - Each parameter-value pair is separated by '=' sign.
  - Series of parameter-value pairs are separated by '&'
  - To use the special character in the URL, use the escape character '%'

  For more information about query strings, see the w3c.org specifications for URL and URI.

- **Name-masking—** You can use name-masking to specify some variables in certain list actions. A name-masked value is not explicit, because it includes a full (*) or partial (for example, ENV*) wildcard, multiple placeholders (%), or both. When placeholders are used in a URI/URL, they need to be "escaped" using the % character. For example, if used in a URI/URL they need to be: %%}

  For wildcarding examples, see Inventory Actions (see page 254).

  Name-masking is only supported on the variables shown in the following table:

List action	Variable
List Packages	Package name
List System	Environment, Stage, System
List Subsystems	Environment, Stage, System, Subsystem
List Element	Environment, Stage, System, Subsystem, Type, Element

  For more information about name-masking, see Name Masking, in the *SCL Reference Guide*.

Repository Actions

Repository actions let you list available instances of CA Endevor SCM. Each instance is defined on the server using a configuration file. Each configuration file defines a Started Task (STC), which points to a specific CA Endevor SCM instance on the Mainframe. For more information about configuration files, see *How Access to a Data Source Works* (see page 218) and *How to Create a Configuration File* (see page 219).

List All Configurations

This action lists all the available CA Endevor SCM configurations on the server.

URL	*<protocol>://<host>:<portnumber>/EndevorService/rest/*
Method	GET
Parameters	None
Output	List of available CA Endevor SCM instances that are defined by configuration files on the the server in JSON format.

For more information about configurations, see How to Enable STC Definitions (see page 195).

List Parameters of a Configuration

This action lists all the parameters of a specific CA Endevor SCM configuration.

URL	*<base URL>/<instance>*
Method	GET
Parameters	None
Output	List of a specific CA Endevor SCM configuration in JSON format.

For more information about configurations, see How to Enable STC Definitions (see page 195).

Package Actions

The RESTful API can process package actions. For more information about package actions and parameters, see the chapter "Managing Packages in Batch," in the *SCL Reference Guide*.

List Packages

The List package action lists CA Endevor SCM packages. Name-masking is supported to filter package names.

URL	*<base URL>/<instance>/packages*
Method	GET

Parameters

status

Filters the list by package status. Specify one or more of the following values: INEDIT, INAPPROVAL, APPROVED, INEXECUTION, EXECUTED, COMMITTED, DENIED.

Example:

<base URL>/<instance>/packages?status=INEDIT&status=EXECUTED

type

Filters the list by package type. Allowed types:

- S - Standard
- E - Emergency

enterprise

Filters the list by enterprise package parameter.

- A - All
- E - Enterprise
- X - Exclude

promotion

Filters the list by promotion package paramter.

- A - All
- P - Promotion
- X - Exclude

Example:

<base URL>/<instance>/packages?status=INEDIT&type=E&enterprise=E

Output

List of Packages in JSON format, which corresponds to the CA Endevor SCM List Package ID function of the CSV utility. For more information, see The List Package ID Function, in the chapter "Using the Comma Separated Value (CSV) Utility," in the *Utilities Guide*.

Notes

To filter by package names use the actual package names or wild carded the package names as the next resource locator as follows:

<base URL>/<instance>/packages/<package_names>

Update Package

The Create/Update package action creates a new package or updates an existing one. If you use this action to update an existing package, the package must be in In-edit status. To use this action, specify a PUT method and use URL with the package name at the end.

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.

URL	*<base URL>/<instance>/packages/<package>*
Method	PUT
Parameters	**ewfromdate=DDMMMYY, ewfromtime=HH:mm, ewtodate=DDMMMYY, ewtotime=HH:mm** (Optional) Specify the time frame within which to execute the package. **Note:** If you specify the ewfromdate, you must also specify the ewfromtime. If you specify the ewtodate, you must also specify the ewtotime. **Example:** *ewfromdate=01JAN00&ewfromtime=20:00&ewtodate=10JAN00&ewtotime=23:59* **type=S/E** (Optional) Specify the package type, where S = STANDARD and E = EMERGENCY. If you do not specify this option and you are creating a new package, the package defaults to a STANDARD (S) package. **sharable=yes/no** Specify whether this package can be edited by more than one person when in In-edit status. If you do not specify this option and you are creating a new package, the package defaults to a NONSHARABLE (no) package. **backout=yes/no** Indicates whether you wish to have the backout facility available for this package. Use this clause when creating a new package only. **Default:** yes. **append=yes/no** Indicates whether to append the SCL you are adding to the existing package SCL or to replace it. **Default:** no.

promotion=yes/no

Define the package as a promotion package or a nonpromotion package. If you do not specify this option and you are creating a new package, the package defaults to a NONPROMOTION (no) package.

fromPackage=

Directs the Create/Update 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.

fromDSN=

fromMember=

Direct the Create/Update action to copy the SCL from the member (fromMember) in the data set name (fromDSN) you specify into the package you are creating or updating. They need to be specified together.

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.

do-not-validate=true

Specify this option to not validate the package components while creating or updating a package.

Output JSON-formatted execution response

Cast Package

The CAST PACKAGE 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. You can use a fully specified, partially wildcarded, or fully wildcarded package ID.

URL *<base URL>/<instance>/<package>/packages/<package>/***Cast**

Method PUT

Parameters **ewfromdate=DDMMMYY, ewfromtime=HH:mm, ewtodate=DDMMMYY, ewtotime=HH:mm**

Change the execution window of the package as part of cast processing.

Note: If you specify the ewfromdate, you must also specify the ewfromtime. If you specify the ewtodate, you must also specify the ewtotime. You can use this clause only if the package ID is fully qualified.

Example:

ewfromdate=01JAN00&ewfromtime=20:00&ewtodate=10JAN00&ewtotime=23:59

validate-components=yes/no/warn

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.

Values:

- YES - enable component validation
- NO - disable component validation
- WARN - generates a warning if component validation fails

backout=yes/no

Indicates whether the backout facility will be available for this package.

Default: yes.

Output	JSON-formatted execution response

Approve Packages

To approve a package specify a package add "Approve" to the end of the url in the following format:

URL	*<base URL>/<instance>/packages/<package>/***Approve**
Method	PUT
Parameters	None
Output	Execution response

Deny Package

To deny a package, specify the package, and add "Deny" to the end of the url in the following format:

URL	*<base URL>/<instance>/packages/<package>/***Deny**
Method	PUT
Parameters	None
Output	JSON-formatted execution response

Execute Package

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. The EXECUTE PACKAGE action support fully specified, partially wildcarded, or fully wildcarded package IDs.

URL	*<base URL>/<instance>/packages/<package>/<package>/***Execute**
Method	PUT
Parameters	**ewfromdate=DDMMMYY** If you specify the *ewfromdate*, you must also specify the *ewfromtime*. **ewfromtime=HH:mm** **ewtodate=DDMMMYY** If you specify the *ewtodate*, you must also specify the *ewtotime*. **ewtotime=HH:mm** Specifies the time frame within which to execute the package (Execution window). You can only use the execution window parameters if the package is fully qualified and the existing execution window is closed. **Example:** *ewfromdate=01JAN00&ewfromtime=20:00&ewtodate=10JAN00&ewtotime=23:59* **status=** Specifies the statuses of the package 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. Valid status values: APPROVED, EXECFAILED
Output	JSON-formatted execution response

Backout Package

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.

URL	*<base URL>/<instance>/packages/<package>/***Backout**
Method	PUT

Parameters	**statement=<statement number>&element=<elementname>** 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 these parameters. For each element action you want to back out, a separate BACKOUT package action with package, statement and element parameters must be provided. **Format:** 1 - 65535. Elementname doesn't support long-named elements.
Output	JSON-formatted execution response

Backin Package

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.

URL	*<base URL>/<instance>/packages/<package>/***Backin**
Method	PUT
Parameters	**statement = <statement number>** **element = <elementname>** 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 these parameters. For each element action you want to back in, a separate BACKIN package action with package, statement and element parameters must be provided. **Format:** 1 - 65535. Elementname doesn't support long-named elements. elements.
Output	JSON-formatted execution response

Commit Package

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.

URL	*<base URL>/<instance>/packages/<package>/***Commit**
Method	PUT

Parameters	**olderthan= number of days** 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. **delete-promotion-history=true** Deletes all the promotion history associated with previous versions of the package.
Output	JSON-formatted execution response

Ship Package

The SHIP PACKAGE action is used 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.

URL	*<base URL>/<instance>/packages/<package>/***Ship**
Method	PUT
Parameters	**destination=** Indicates the name of the remote site to which you want to ship the specified package. This parameter is mandatory. **Format:** 1 to 8alphanumeric character site name. Must begin with an alphabetic character **Examples:** DESTINAT, D123 **option=** Indicates whether you want to ship output members, or backout members to the remote site. **Examples:** OUTPUT or BACKOUT **prefix=** Indicates the data set name prefix to be used in the XCOM or CONNECT:DIRECT transmission methods. **Format:** 1 to 8 characters
Output	JSON-formatted execution response

Delete Package

The DELETE PACKAGE action allows you to delete packages. You can use the DELETE PACKAGE action to delete packages of any status type.

URL	*<base URL>/<instance>/packages/<package>*

Method	DELETE
Parameters	**olderthan= number of days** 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. Optional for wild carded package name and ignored when package name is fully specified. **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. **Valid status values:** ALLSTATE, INEDIT, INAPPROVAL, APPROVED, INEXECUTION, EXECUTED, EXECFAILED, COMMITTED, DENIED
Output	JSON-formatted execution response

Reset Package

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.

URL	*<base URL>/<instance>/packages/<package>/***Reset**
Method	PUT
Parameters	None
Output	JSON-formatted execution response

Inventory Actions

Inventory actions let you list Systems, Subsystems, and Elements for a particular inventory location. The parameter string must precisely define the sequence of parameters that correspond to the inventory structure. The sequence is static, meaning that each parameter has a defined position and must be specified, either explicitly or using an astrick (*), if it is required for the action.

The inventory sequence string is attached to the <base URL>/<instance> to form the full request URL.

Inventory sequence string

```
/<environment>/<stage number>/<system>/<subsytem>/<type>/<element name>
```

The sequence defines a resource in CA Endevor SCM and when evaluated from left to right it defines the resource from general to specific.

Examples: Inventory Action Sequence Strings

- http://ndvrserver:8080/EndevorService/rest/DEMONDVR/*/*/*/*

  This URL will be used to list Subsystems, where all parameters are wild carded and DEMONDVR is the CA Endevor SCM configuration name.

  Environment=*, stage number= *, system = * and subsystem = *

- http://ndvrserver:8080/EndevorService/rest/DEMONDVR/ENV1/1/DEMOSYS

  This URL will be used to list Systems DEMOSYS, Environment is ENV1 and Stage number is 1.

- http://ndvrserver:8080/EndevorService/rest/DEMONDVR/ENV1/1/DEMOSYS/DEMOSBS/ASMPGM/LONG*

  This URL will be used to list Elements that match the wild carded pattern LONG*, Environment=ENV1, Stage number=1, System=DEMOSYS, Subsystem=DEMOSBS and Type=ASMPGM.

List Systems

The list System function allows you to extract System information from the Master Control File that satisfies the criteria you specify.

URL	*<base URL>/<instance>/<environment>/<stage numbers>/<system>*
Method	GET
Parameters	**path = LOG/PHY** PATH LOGICAL and PATH PHYSICAL option **search = NOS/SEA** SEARCH and NOSEARCH option **return = FIR/ALL** RETURN FIRST or RETURN ALL option The PATH, SEARCH, NOSEARCH and RETURN clauses can be omitted. Default mapping options on non-explicit Environment name requests are: PATH PHYSICAL, NOSEARCH and RETURN ALL. The PATH LOGICAL, SEARCH and RETURN FIRST clauses are invalid on non-explicit Environment name requests. Default mapping options on explicit Environment name requests are: PATH LOGICAL, RETURN FIRST and NOSEARCH. Name-masking is allowed for all parameters (Environment, Stage, System). CA Endevor SCM has a limitation regards length of Environment, System length up to 8 characters. Parameter Stage is expected to be a Stage ID.

Output	List of Systems in JSON format, which corresponds to the CA Endevor SCM List System function of the CSV utility. For more information, see The List System Function, in the chapter "Using the Comma Separated Value (CSV) Utility," in the *Utilities Guide.*

List Subsystems

The list Subsystem function allows you to extract Subsystem information from the Master Control File that satisfies the criteria you specify.

URL	*<base URL>/<instance>/<environment>/<stage numbers>/<system>/<subsystem>*
Method	GET
Parameters	**path = LOG/PHY** PATH LOGICAL and PATH PHYSICAL option **search = NOS/SEA** SEARCH and NOSEARCH option **return = FIR/ALL** RETURN FIRST or RETURN ALL option The PATH, SEARCH, NOSEARCH and RETURN clauses can be omitted. Default mapping options on non-explicit Environment name requests are: PATH PHYSICAL, NOSEARCH and RETURN ALL. The PATH LOGICAL, SEARCH and RETURN FIRST clauses are invalid on non-explicit Environment name requests. Default mapping options on explicit Environment name requests are: PATH LOGICAL, RETURN FIRST and NOSEARCH. Name-masking is allowed for all parameters (Environment, Stage, System, Subsystem). CA Endevor SCM has a limitation regards length of Environment, System, and Subsystem length up to 8 characters. Parameter stage is expected to be a Stage ID.
Output	List of Subsystems in JSON format, which corresponds to the CA Endevor SCM List Subsystem function of the CSV utility. For more information, see The List Subsystem Function, in the chapter "Using the Comma Separated Value (CSV) Utility," in the *Utilities Guide.*

List Elements

The list Elements function lets you extract Element information from the Master Control File that satisfies the criteria you specify.

URL	*<base URL>/<instance>/<environment>/<stage number>/<system>/<subsystem>/<type>/<elementname>*
Method	GET

Parameters

data = ALL/BAS/ELE

(Optional) If not specified, the default is DATA ALL. However, DATA ALL does not return change level summary data. DATA BAS writes basic summary and DATA ELE writes Element summary.

path = LOG/PHY

PATH LOGICAL and PATH PHYSICAL option

search = NOS/SEA

SEARCH and NOSEARCH option

return = FIR/ALL

RETURN FIRST or RETURN ALL option

The PATH, SEARCH, NOSEARCH and RETURN clauses can be omitted. Default mapping options on non-explicit Environment name requests are: PATH PHYSICAL, NOSEARCH and RETURN ALL. The PATH LOGICAL, SEARCH and RETURN FIRST clauses are invalid on non-explicit Environment name requests. Default mapping options on explicit Environment name requests are: PATH LOGICAL, RETURN FIRST and NOSEARCH.

Name-masking is allowed for all parameters (Environment, Stage, System, Subsystem, Type, Element). CA Endevor SCM has a limitation regards length of Environment, System, and Subsystem length up to 8 characters. Parameter Stage is expected to be a Stage ID.

Output

List of Elements in JSON format, which corresponds to the CA Endevor SCM List Element function of the CSV utility. For more information, see The List Element Function, in the chapter "Using the Comma Separated Value (CSV) Utility," in the *Utilities Guide*.

Chapter 7: Best Practice Implementation

The following information describes how a CA Endevor SCM administrator can set up a complete implementation of the product using the Best Practice Implementation (BPI) method. The BPI defines a best practices lifecycle for active software development. The BPI process allocates all the necessary files and sets up all the definitions that are required to allow you to begin to use the product and is delivered as a set of input tables and jobs that you configure and run. After you complete the BPI, you will be ready to load your source code into the CA Endevor SCM inventory structure that supports the software development lifecycle.

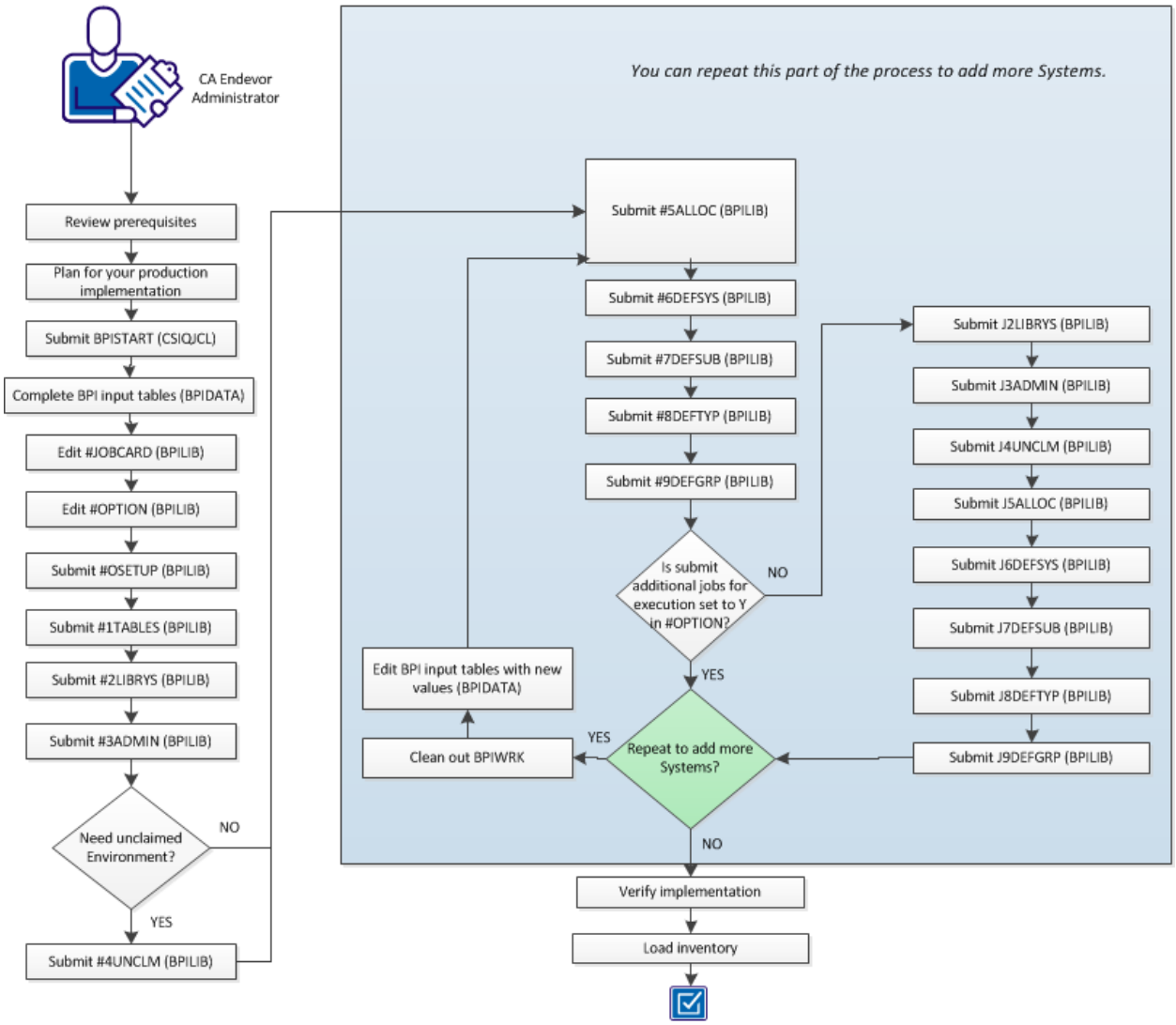
This section contains the following topics:

How to Perform the Best Practice Implementation

As a change manager (CA Endevor SCM administrator), you can use the Best Practice Implementation (BPI) to set up a complete implementation of CA Endevor SCM that incorporates best practices. This method enables you to easily and quickly allocate and populate the libraries that support the software development lifecycle. The BPI is delivered as a set of input tables and jobs that you configure and run.

The following diagram outlines the Best Practice Implementation process:

How to Perform a Best Practice Implementation



The Best Practice Implementation (BPI), includes the following steps:

1. Review the prerequisites. (see page 262)
2. Plan for your implementation. (see page 263)
3. Run BPISTART. (see page 277) This job allocates the BPI libraries.
4. Customize the BPI input tables. (see page 278)
5. Edit #JOBCARD to prepare the BPI job card. (see page 288)
6. Edit #OPTION. (see page 288) This member defines symbol values that are used to resolve variables defined in the BPI models and REXX members.
7. Submit the BPI jobs. (see page 289) We recommend that you execute the BPI process in iterations implementing one System or application at a time.
8. Verify the implementation. Access your implementation and make sure it was created and setup successfully.
9. Load your software inventory. We recommend that you complete setting up the BPI, before you load any inventory.

We recommend that you execute the BPI process in iterations. For the first iteration, you might implement one System (or application). This lets you become familiar with how to edit the input tables and how the jobs work. After the first iteration, you could execute the BPI to implement an entire application. (An application can consist of multiple Systems and Subsystems.) Then you could do additional iterations, one for each additional application. For the first iteration, you perform all the steps. For subsequent iterations, you re-edit the BPI input tables and then rerun the jobs beginning with job #5ALLOC. For more information about performing the steps in iterations, see The Iterative BPI Process (see page 290).

Review Prerequisites

Before attempting the Best Practice Implementation (BPI), verify that the following prerequisites are complete.

Verify that CA Endevor SCM is properly installed at your site. For more information about the installation steps, see the Installation Tasks for the Systems Programmer in the "Overview" chapter in the *Installation Guide*. Proper installation has the following requirements:

- The CA Common Services components CA LMP and CAIRIM for product licensing and authorization must be installed and customized.

  Note: The components CAIENF and CAICCI are not required for the BPI. However, these components are required if you want to use CA Endevor SCM with the Concurrent Action Processing feature, the Web Services component, or the companion product CA CMEW. CAIENF and CAICCI can be configured after you complete the BPI.

- NDVRC1 must be authorized.
- BC1JJB03 must have been executed.
- The load modules contained in *iprfx.iqual*.CSIQAUTH and *iprfx.iqual*.CSIQAUTU (and optionally *iprfx.iqual*.CSIQLOAD) must reside in a system-authorized library. You can use one of the following methods to make sure that the load modules are in a system-authorized library:
  - Copy the members from *iprfx.iqual*.CSIQAUTH and *iprfx.iqual*.CSIQAUTU libraries to an existing authorized library.
  - Authorize *iprfx.iqual*.CSIQAUTH and *iprfx.iqual*.CSIQAUTU.
  - Copy members from *iprfx.iqual*.CSIQAUTH and *iprfx.iqual*.CSIQAUTU (and optionally *iprfx.iqual*.CSIQLOAD) to an existing authorized LINKLIST library.
  - Copy members from *iprfx.iqual*.CSIQAUTH and *iprfx.iqual*.CSIQAUTU (and optionally *iprfx.iqual*.CSIQLOAD) to an existing LPA library.
  - Define the *iprfx.iqual*.CSIQAUTH and *iprfx.iqual*.CSIQAUTU (and optionally *iprfx.iqual*.CSIQLOAD) data sets as LINKLIST or LPA libraries themselves.
- Define CA Endevor SCM to the ISPF Environment.
- If your site has multiple CPUs sharing DASD, you must define queue names. If you are using Unicenter CA-MIM Resource Sharing or IBM Global Resource Serialization (GRS), include the queue names in the appropriate Global Resource Queue Name Table.
- If CA ACF2 is the security software used at your site, modify the TSO Command Limiting Facility to allow ISPTLIB commands (ACMQ, BC1PACMI, ESORT, EONLY, EM, QM, and so on).

The following BPI installation considerations apply depending on how CA Endevor SCM is configured at your site:

- The Best Practice Implementation assumes your site is licensed for the following product options:
  - CA Endevor Extended Processors
  - CA Endevor Automated Configuration
- All steps described in the chapter "Configuring Your Product" in the Installation Guide must be performed. The following exceptions apply to the steps in the Configuring Your Product chapter:
  - If you use LINKLIST or LPA libraries for the CSIQAUTH/CSIQAUTU data sets, you will need to tailor the jobs created by the BPI process prior to submission. Remove, or comment out, the STEPLIB DD statements from the job members created in the BPILIB library before submitting the job.
  - If you use LINKLIST or LPA libraries for the CSIQLOAD data set, you will need to tailor the jobs created by the BPI process prior to submission. Remove, or comment out, the STEPLIB and CONLIB DD statements from the job members created in the BPILIB library before submitting the job.

The person performing the BPI needs the following general knowledge of the mainframe environment and a basic understanding of the CA Endevor SCM.

- You need a working knowledge of the mainframe environment, the z/OS mainframe operating system, the Time Sharing Option facility (TSO), and the Interactive System Productivity Facility (ISPF).
- Become familiar with the basic CA Endevor SCM concepts. For more information, see the *Installation Guide* and the *Administration Guide*.
- Become familiar with the Best Practice Implementation concepts. For more information about the BPI concepts, see the BPI Software Inventory Lifecycle (see page 264).

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

Plan for Your Production Implementation

To perform the Best Practice Implementation (BPI) process, you will need to decide how you want to setup your CA Endevor SCM configuration. The BPI automates some of those decisions by incorporating "best practices" and recommended settings automatically within the process. It is important to know and understand the sort of implementation created when performing the BPI process. This section describes each of those best practices in more detail.

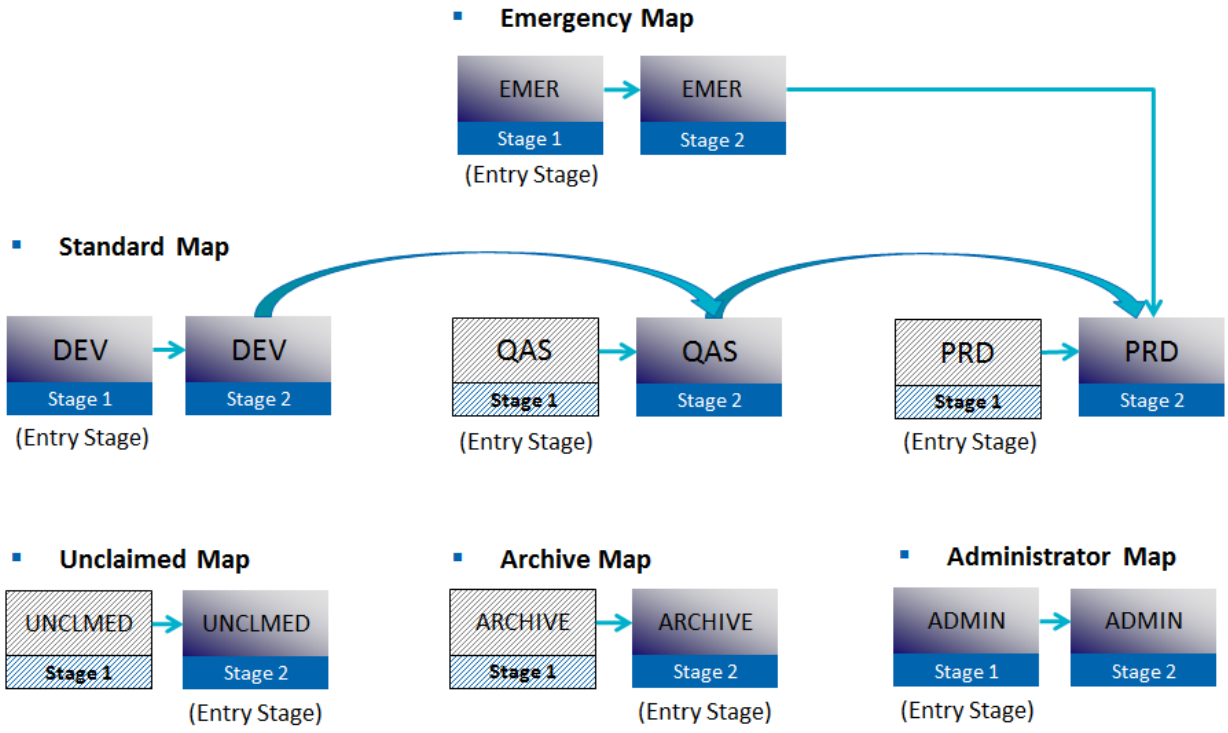
BPI Software Inventory Lifecycle

The Best Practice Implementation sets up a seven-environment mapping structure, by default. This structure includes the Development (DEV), Quality Assurance (QAS), Production (PRD), Emergency (EMER), Unclaimed (UNCLMED), Archive (ARCHIVE), and Administrator (ADMIN) environments. Your site may not need all seven environments, or you may need more than seven. However, this represents an active development lifecycle that is appropriate for most sites.

The lifecycle defines the movement of your software inventory through a set of data set libraries. These lifecycles are defined in the Defaults table (C1DEFLTS). A data set naming convention supports the lifecycle structure.The Best Practice Implementation sets up the following lifecycles or maps:

- Standard Map— For regular development activities.The Standard Map is where most activity will take place. The Standard Map starts with the DEV environment, at entry stage, stage 1. From DEV Stage 1, source will be promoted to DEV Stage 2, then to QAS stage 2, and finally PRD stage 2. Stage one for both the Quality Assurance and Production environments are skipped. DEV stage 2 maps to QAS stage 2, and QAS stage 2 maps to PRD stage 2.
- Emergency Map— For emergency fixes. The Emergency environment maps also to PRD stage 2.
- Unclaimed Map— For unclaimed or unclassified elements. The Unclaimed environment has an entry stage of 2. This environment does not map to any other environment.
- Archive Map— For archived elements. The Archive environment has an entry stage of 2. This environment does not map to any other environment.
- Administrator Map— For the CA Endevor SCM Administrator activities. The Administrator environment has an entry stage of 1. This environment does not map to any other environment.

The following graphic shows how the BPI environments are mapped:



BPI Environment and Stages

Each lifecycle defines the movement of software inventory from environment-stage to environment-stage until the end of the lifecycle. Each environment has two stages and each environment-stage represents a library level. To support this structure, the data set naming convention uses qualifiers to identify each environment and each stage. Details follow about each environment.

- **Development (DEV) environment—** This environment has two stages that allow you to maintain two versions of an element. You can use Stage 2 as a location for elements that are considered complete or as a temporary holding place for confirmed changes, while you re-edit the element in Stage 1. Stage 2 also provides another location useful for merging changes together with those that have been promoted to QAS or PRD. Another benefit of two development stages is to be able to test with different compiler parameters at Stage 1 verses Stage 2. For example, Stage 1 could be set to test compile options and Stage 2 could be set to production compile options. The last compile, before moving up the lifecycle, is always done at DEV Stage 2.
- **Quality Assurance (QAS) environment—** This environment is used to perform many different types of testing. Stage 1 provides a common integration stage. Although useful, to reduce the number of stages an element must pass during the path to production, Stage 1 of the QAS environment is not used in the BPI delivered lifecycle. Stage 2 would be used to make sure the applications are system tested and ready for production.

- **Production (PRD) environment—** This environment represents the production code. Stage 1 of the PRD environment is not used in the BPI delivered lifecycle. Stage 2 is used for the storage of the production source. The PRD environment should be package protected.
- **Unclaimed (UNCLMED) environment—** This environment is used to store elements that could not be classified in the standard inventory structure (systems, subsystems, and so on.). Only the second stage is used for this environment. This environment is intended to be a holding area for elements whose use and purpose is unknown. This may be common in conversion situations, when converting from a different source control management application to CA Endevor SCM. The UNCLMED environment stores only the source and has only one system and subsystem. Its purpose is a temporary holding position for those elements not identified with a system during the implementation (or conversion) process. If an element is found missing after the CA Endevor SCM Implementation is complete, you can look in the UNCLMED environment to see if the element is there. You can then transfer that element from UNCLMED to the appropriate system and subsystem. After elements in UNCLMED have been identified and transferred into the correct system-subsystem, the version in UNCLMED should be deleted. The final goal should be to discontinue the UNCLMED environment in time.
- **Administrator (ADMIN) environment—** This environment is separate from the DEV to QAS to PRD lifecycle, and is reserved for the role of the CA Endevor SCM administrator. The CA Endevor SCM administrator has different security requirements than the development and testing teams. The separate security rules for the administrator role are easier to maintain if in a separate ADMIN environment. The ADMIN environment's data sets begin with a different high level qualifier, while the remaining environments (DEV, QAS, PRD, EMER, UNCLMED, and ARCHIVE) all use the application high level qualifier. Stage 2 of the ADMIN environment hosts the CA Endevor SCM Configuration Tables that are used to define the site's infrastructure and default options, as well as the Endevor processors which are used to process all the elements for each system in the standard DEV to QAS to PRD lifecycle. Stage 1 of the ADMIN environment provides an area for the administrator to make changes as needed to any of the tables or processors maintained within the ADMIN environment. Stage 2 only stores the elements specific to CA Endevor SCM administrators. Sites can then use packages to promote their changes from Stage 1 to Stage 2, allowing for a back-out if needed.

- **Emergency (EMER) environment—** This environment is also separate from the standard lifecycle. This environment is used for coding fixes that need to be put in immediately. Stage 1 of the EMER environment serves as the place where the code changes are to be made. Stage 2 of EMER is package protected and must be included in the production JCL STEPLIB concatenation. An emergency package needs to be created to promote elements from Stage 1 to Stage 2. An approval needs to be made by those that were assigned in the approver group(s). An emergency approver group must be given the authority to approve emergency packages. This is required to prevent unapproved changes being picked up by production. EMER maps to the PRD environment for ease of making changes to programs and components. It is probable that emergency changes promoted to EMER Stage 2 will afterwards be transferred to the DEV environment (via the TRANSFER action) and then promoted through the typical lifecycle and testing procedures before being permanently moved to production. The EMER environment contains the same types, systems, and subsystems as the PRD environment. The EMER Stage 2 loadlib should be concatenated before the production stage 2 loadlib, to pick up any emergency changes.
- **The Archive (ARCHIVE) environment—** This environment is used to store elements that are deleted from the PRD environment, but a site still wants to maintain a copy of for auditing or history purposes. The ARCHIVE environment utilizes only the second stage of this environment (Stage 2). Elements need to be transferred there via the TRANSFER action or via a PACKAGE with TRANSFER action SCL. It is a stand-alone environment. The ARCHIVE environment contains all the types, systems, and subsystems of the PRD environment.

Stage Name, Number, ID

When defining Stage names, numbers, and IDs, it is best to establish a simple pattern. Each Environment contains two Stages, regardless of whether you use both Stages, and the Stage numbers are 1 and 2. With the BPI, the Stage ID is the same as the Stage number, and Stage names are built using an easy to remember naming convention. As long as the Environment name is less than eight characters, the Stage name can be a combination of the Environment name and the Stage ID or number.

Data Set Naming Conventions

Having standard naming conventions within various components of CA Endevor SCM makes the product easier to support. The Best Practices Implementation (BPI) method includes standard data set naming conventions.

Previously if you were not using the BPI method, CA Endevor SCM used two sets of high and second level qualifiers for data set naming (iprfx.iqual and uprfx.uqual). The first set was reserved for the CA Endevor SCM installation runtime data sets. These data sets were prompted for during the installation process (job BC1JJB03). The second set was used as the prefix for *all* the CA Endevor SCM data sets that were created (element catalog, MCFs, package file, base and delta files, processor output libraries, and so on).

When you configure CA Endevor SCM using the BPI, you must specify four high level qualifiers for the CA Endevor SCM data sets. The installation runtime qualifier (iprfx.iqual) remains the same. The remaining three high level qualifiers are specified in the BPI Option member. Having these multiple high level qualifiers makes it easier for administrators to establish and maintain security rules, as well as provide a visual distinction between the data sets. The three prefixes are as follows:

- SiteHighLevelQual for all CA Endevor SCM site-wide data sets. This includes element catalog and index, ACM root and cross reference files, package file and Master Control files.
- AdminHighLevelQual for all CA Endevor SCM ADMIN environment data sets. This includes all allocations for the ADMIN environment.
- ApplHighLevelQual for all CA Endevor SCM application data sets. This includes the base and delta files for each of the standard application system, subsystem, and type combinations.

The maximum size for a data set name cannot exceed 44 characters, thus the maximum sizes allowed for the three high-level qualifiers will depend on the system and subsystem names. You can use multiple nodes within each high-level qualifier. The number of nodes is not important. However, the CA Endevor SCM data set names use a standard format consisting of the following, where the maximum number of characters allowed are shown in parenthesis: system (8), subsystem (8), subset of the environment name (3), stage number (1) and type (8). Using this standard, if you use the maximum amount of characters for system, subsystem, and so on, then only 12 characters are left for the high-level qualifier nodes. We recommend that the prefix nodes be 12 characters or fewer, if possible. This is to avoid data set length problems in the future if additional systems, subsystems, and types are created.

Data Set Rules

When allocating data sets for the BPI method, follow these rules:

- One base library per environment, system, subsystem, stage, type.
- Type name is the last node of the data set name and reflects the language.
- Base libraries are defined as PDSEs.
- One delta library per environment, system, subsystem.
- Provide High Level Qualifier only, the BPI provides an ending qualifier. Processors also use this naming convention.
- Reverse delta format is used for most types. Some types are better suited for full image or log delta format.

Centralized Processors

The Best Practice Implementation establishes a single repository location for processors instead of having unique or custom processors defined and stored for each system. It is simpler to maintain one set of processors and ensures consistency across systems.

The BPI process stores all processors in the administrator (ADMIN) environment.

Inventory Structure

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

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

With the BPI, a predefined inventory structure is included. The structure (environments, systems, subsystems, types, and so on) is outlined in rows within table file members.

Understanding the Inventory Structure components

The BPI provides the following default definitions for the inventory structure components.

Environments

The default environment names are delivered in the BPI input table, T#ENVMTS. Each data row consists of the environment name and stage information.

Note: BPI REXX member, O#ENVMTS located in BPILIB must be kept in sync with T#ENVMTS.

The rows in these tables are used to build the software change lifecycle (established within the C1DEFLTS table), the CA Endevor SCM configuration tables, and to build SCL members. You specify what environments you want to create, the environment stage names, and the entry stage for that environment. Also, you specify how the these are mapped— the next environment to map to and the next stage of that environment to map to.

Systems

The default system names are delivered in members T#ADMSYS and T#SYSTMS. Each data row consists of a system name and a description. T#ADMSYS is used solely for the ADMIN environment. The ADMIN environment, as delivered, will contain one system and subsystem. It is required that you keep the administrative environment. The BPI defaults to creating the first ADMIN system specified in T#ADMSUB as the main repository of all the CA Endevor SCM processors and configuration tables that the remaining application systems use. The system name for the Processor Output Libraries will default to the first system specified in T#ADMSYS. You may choose to add additional systems to the ADMIN environment, by adding additional rows to the T#ADMSYS table.

T#SYSTMS table is used by all the environments defined in the T#ENVMTS table except for Administrator and Unclaimed environments. That includes the Development, Quality Assurance, Production, Emergency, and Archive environments in the delivered table.

This T#SYSTMS table is related to the T#SUBSYS table. If you add or remove a system row from the T#SYSTMS table, you must add or remove the associated row(s) in the T#SUBSYS table.

The system definition defaults delivered with the BPI are listed below:

- Next System – defaults to the current system
- Comments Required
- CCID Required
- Duplicate Element Check is Not Active
- Duplicate Processor Output Check is Active— Severity Level is ‘E’
- Element Jump Acknowledgment Required
- Signout is Active
- Signout Dataset Validation is Not Active

Subsystems

The default subsystem names are delivered in members T#ADMSUB and T#SUBSYS. Each row consists of a system name, subsystem name and a description. Any system specified in T#SUBSYS must match a system specified in T#SYSTMS. Any system specified in T#ADMSUB must match a system specified in T#ADMSYS

The delivered subsystem definition specifies that the Next Subsystem defaults to the current subsystem.

Types

The default type names are delivered in members T#ADMTYP and T#TYPES. The T#ADMTYP member is used to define types for the Administrator environment and the T#TYPES member is used to define types for the Development, Quality Assurance, Production, Emergency, Unclaimed, and Archive environments. The length of the data in each row of these members is 200 characters. Each row consists of a type name, processor group name, generate, move and delete processor names, foreground allowed flag, language, panvalet and librarian source languages, compare from and compare to columns, element delta format, and type and processor group descriptions. When a type is defined during the BPI process, the first processor group listed for each type will be the default processor group.

Note: If the BPI process is executed and your site did not purchase the Extended Processor option, the implementation will fail at the point when the DEFINE TYPE PROCESS action is executed. The execution of this action will result in a return code of 12 when CA Endevor SCM performs a check against the PROC parameter value in the C1DEFLTS configuration table and determines this option is not activated.

The type definition defaults delivered with the BPI are listed below:

- Do Not Expand Includes
- Element Delta Format is Reverse
- Do Not Compress Base
- Regression Threshold is 80
- Regression Severity is Caution
- Do Not Consolidate Element Levels
- Consolidate Elements at Level 96
- Number of Element Levels to Consolidate 0
- Do Not Consolidate Components Levels
- Consolidate Components at Level 96
- Component Delta Format Reverse
- Number of Component Levels to Consolidate 0
- HFS Recfm NL

Note: The Regression Percentage Threshold is automatically set to 80, except for LOG or IMAGE delta formats, where it is set to 0.

Processor Groups

The processor group and processor names are included in members T#ADMTYP and T#TYPES along with the type attributes.

One exception to the naming standard is for the special Endevor processor group used by the type PROCESS. The name of this processor group is PROCESS.

The processor group definition defaults delivered with the BPI are listed below:

- Move Action Uses Move Processor
  - If the entry stage of the environment is stage 1, then it will use the Generate processor. The production environment will always use the Move processor
- Transfer Action Uses Move Processor
- Allow or Do Not Allow Foreground Execution
  - The Allow or Do Not Allow is specified in the T#ADMTYP and T#TYPES table. The Unclaimed and Archive environments will use Allow for all types
- Processor Output Type
- Generate/Move/Delete Processors
  - The processor names will be taken from the T#TYPES table, except for type PROCESS, which will always use the Endevor processor names. The Unclaimed and Archive environments set all processor values to *NOPROC* for all types

Processor Symbols

Processor group symbol overrides are created in #3ADMIN and #9DEFGRP, to coincide with the delivered BPI processors in CSIQSAMP. There are two tables within the #3ADMIN JCL that define what symbols need to be created for the administrator environment. The standard environments use the T#SYMBLS table. Processor symbols can be a length of eight characters and the override values can be up to 65 characters. As delivered, there are processor symbol overrides required since a single processor is shared across multiple processor groups. You will have the flexibility to change or add to the delivered processor symbol definitions by editing #3ADMIN or T#SYMBLS.

The processor symbols definition defaults delivered with the BPI are listed below:

Processor Symbols Definition Defaults

The processor symbols definition defaults delivered with the BPI are listed below:

Symbol	Override	Location	Table
HLQ	&#HLQADM	Administrator Environment, all systems and processor groups, for GEN and MOVE	#3ADMIN

XSYSLIB1	&C1MACLIB	Administrator Environment, all systems, type TABLE, procgrp ASMTBL, stage 1 & 2, for GEN only	#3ADMIN
LOADLIB	&CSIQAUTU	Administrator Environment, all systems, type TABLE, procgrp ASMTBL, stage 2, for GEN only	#3ADMIN
LOADLIB	&CSIQAUTU	Administrator Environment, all systems, types ASMPGM and COBOL, procgrp USEREXIT, stage 2, for GEN only	#3ADMIN
LANGVER	COBII	Administrator Environment, all systems, type COBOL, procgrp BATCHCII, stage 1 & 2, for GEN only	#3ADMIN
LANGVER	COBII	All environments except Administrator, Archive, and Unclaimed, all systems, type COBOL, procgrp BATCHCII, CICSCII, DB2CII, and IDMSCII, stage 1 & 2, for GEN only	T#SYMBLS
IDMSDBN	SYSTEM	All environments except Administrator, Archive, and Unclaimed, all systems, type COBOL, procgrp IDMSCII and IDMSCLE, stage 1 & 2, for GEN only	T#SYMBLS
IDMSDICT	SYSDICT	All environments except Administrator, Archive, and Unclaimed, all systems, type COBOL, procgrp IDMSCII and IDMSCLE, stage 1 & 2, for GEN only	T#SYMBLS
IDMSDMCL	USERDB	All environments except Administrator, Archive, and Unclaimed, all systems, type COBOL, procgrp IDMSCII and IDMSCLE, stage 1 & 2, for GEN only	T#SYMBLS
DDLDML	DBLOADLIB	All environments except Administrator, Archive, and Unclaimed, all systems, type COBOL, procgrp IDMSCII and IDMSCLE, stage 1 & 2, for GEN only	T#SYMBLS
DDLDCLOD	DBLOADDSN	All environments except Admin, Archive, and Unclaimed, all systems, type COBOL, procgrp IDMSCII and IDMSCLE, stage 1 & 2, for GEN only	T#SYMBLS
DDLDCMSG	DBMSGDSN	All environments except Admin, Archive, and Unclaimed, all systems, type COBOL, procgrp IDMSCII and IDMSCLE, stage 1 & 2, for GEN only	T#SYMBLS

Configuration Tables

CA Endevor SCM configuration table source are delivered in the *iprfx.iqual*.CSIQSRC deployment library. New C1DEFLTS and ESYMBOLS configuration table source is built by the BPI process, based on information specified in the table input members located in the BPIDATA data set and information specified in the #OPTION member located in the BPILIB data set. None of the other configuration table source is altered by the BPI process.

- BPI job #1TABLES builds the two new tables and creates load modules for the C1DEFLTS, ENCOPTBL, ENDICNFG and ESYMBOLS tables.
- BPI job #3ADMIN builds an inventory location for the tables (ADMIN/ADMIN/ENDEVOR/TABLE/2) and adds and promotes all the configuration tables to that location.

You can modify the source as required after the implementation is complete.

Note: When performing the Best Practice Implementation, we recommend that you set the WARN parameter to YES in the CA Endevor SCM configuration table, BC1TNEQU. When WARN=YES, the ESI warning mode is in effect. Warning mode allows access to resources even if your site security package (RACF, CA ACF2, or CA Top Secret) indicates that access should be denied. You should use ESI Warning Mode when running the Best Practice Implementation. System Management Facility (SMF) messages will indicate any authorization issues with data sets, but will not halt the BPI implementation process. After running the implementation process, you can adjust your security rules and change the WARN parameter to NO, if you prefer not to run CA Endevor SCM in warning mode. For more information about the WARN parameter, see The ESI Warning Mode in the *Security Guide*.

Global Type Sequencing

By default, the BPI enables Global Type Sequencing. When this feature is enabled, API element actions (if wildcarded or masked) and SCL element actions are processed by a single Type sequence defined at the site level in the Type Sequence member created by the administrator.

Global Type Sequencing is a prerequisite for the following features:

- Concurrent Action Processing (CAP)— When CAP is specified for a batch job or packages submitted for processing, certain element action requests are executed concurrently. CAP uses Global Type Sequencing to determine which actions can be processed at the same time.
- Autogen option for Add, Update, and Generate actions— When Autogen is specified for an element, elements that use this component element are automatically generated. Autogen requires Global Type Sequencing so that the elements that use a component are generated after all of the components have been processed (for example, macros before source, source before load modules).

To enable Global Type Sequencing during the BPI process, you define the type sequence order in the table T#GLBSEQ. Types and System names need to match those you specify in the T#TYPES and T#SYSTMS tables.

Best Practice Implementation Source Files

The CA Endevor SCM Best Practices Implementation (BPI) is delivered as part of the base product. All the BPI components reside in the following deployment libraries along with other CA Endevor SCM base members;

- *iprfx.iqual*.CSIQSAMP—Skeleton, model, and BPI processor members
- *iprfx.iqual*.CSIQCLS0—REXX procedures and REXX options statement members
- *iprfx.iqual*.CSIQJCL—Job control statement members
- *iprfx.iqual*.CSIQDATA—BPI input table members

The iprfx.iqual set of libraries is referred to as the deployment or runtime libraries. These are the libraries that contain the product software that is executed. These members are edited by the BC1JJB03 installation job as part of the installation process.

Run BPISTART

To start the Best Practice Implementation, you edit and submit the job, BPISTART.

Follow these steps:

1. Set the WARN parameter to YES in the CA Endevor SCM configuration table, BC1TNEQU.

   Note: When performing the Best Practice Implementation, we recommend that you set the WARN parameter to YES in the CA Endevor SCM configuration table, BC1TNEQU. When WARN=YES, the ESI warning mode is in effect. Warning mode allows access to resources even if your site security package (RACF, CA ACF2, or CA Top Secret) indicates that access should be denied. You should use ESI Warning Mode when running the Best Practice Implementation. System Management Facility (SMF) messages will indicate any authorization issues with data sets, but will not halt the BPI implementation process. After running the implementation process, you can adjust your security rules and change the WARN parameter to NO, if you prefer not to run CA Endevor SCM in warning mode. For more information about the WARN parameter, see The ESI Warning Mode in the *Security Guide*.

2. Locate BPISTART in the *iprfx.iqual*.CSIQJCL library.

3. (Optional) Edit the SET LBPRFX=u*prfx.uqual* statement in BPISTART to override the uprfx and uqual values specified in the BC1JJB03 installation job.

4. Add a valid job card statement and submit the BPISTART job for execution.

   This job does the following:

   - Allocates the BPI libraries: uprfx.uqual.BPIDATA, uprfx.uqual.BPILIB, uprfx.uqual.BPIPRCR, and uprfx.uqual.BPIWRK.

     Note: These libraries will be modified during the BPI process. The delivered libraries will not be modified and thus will be available for backup if the BPI process needs to be restarted for any reason.

   - Copies delivered and tailored members into the BPIDATA, BPILIB and BPIPRCR libraries.

   The following libraries are created with the uprfx.uqual values specified in the BPISTART job:

   - *uprfx.uqual*.BPIDATA— Contains all the default input table members. This is a copy of the iprfx.iqual.CSIQDATA library. These tables are used to define the inventory structure (environments, systems, subsystems, types, processor groups, and so on.) to the BPI process jobs.

   - *uprfx.uqual*.BPILIB— Contains several members copied from other installation libraries, JCL members built by the BPI process such as job #1TABLES - #9DEFTYP and several other files that are built by the BPI process, including the C1DEFLTS and ESYMBOLS configuration table source. This is the library the user works from to execute the BPI process jobs.

- *uprfx.uqual*.BPIPRCR— Contains the source for the delivered BPI CA Endevor SCM processors. They are added into the Administrator environment by the #3ADMIN BPI job.
- *uprfx.uqual*.BPIWRK— Contains temporary and intermediate members built by one of the BPI jobs or steps and passed to another job or step later in the process. Examples include the SCL built by the process to DEFINE SYSTEMs and SUBSYSTEMs. Other members are placed into this library for support purposes in case any issues arise.

Customize the BPI Input Tables

A default set of BPI tables are copied into the BPIDATA library. These files determine the configuration using predefined best practice values. However, you must edit these tables as appropriate for your site.

Edit each member in the BPIDATA library depending on how you decide to configure your implementation. You can use conversion utilities, the ISPF Editor, upload from a word document, or other methods may be used to modify or create these required tables.

Important: When editing tables, make sure that all changes conform to the table format rules.

Important: How you customize the input tables depends on how many times you plan to execute the BPI process. For more information about planning your implementation, see The Iterative BPI Process (see page 290).

Input Table Format Rules

When you edit an input table, make sure to conform to the following rules.

- Heading row
  - One heading row (single asterisk in column 1) must exist in each table.
  - The heading row must be coded prior to any data rows.
  - The format of the delivered heading rows must not be changed in any way.
- Data rows
  - All values in the data rows must be entered in upper case mode unless otherwise stated.
  - All values are required unless otherwise stated.
  - The data in the data rows must be placed in the same columns as the field literal in the heading row.
- Comment rows
  - Comment rows can be added anywhere within the table source by placing asterisks in columns 1-2 of a row. When detected the row is ignored.

T#ADMALL--Allocations for Admin Environment

The T#ADMALL table allocates data sets for the ADMIN environment. This table enables the creation of the following data sets for each element type listed in the table.

- One DELTA, LISTLIB, LLISTLIB, LOADLIB, and OBJLIB library for each element type for the ADMIN environment.
- One base library for each element type for each system, subsystem, environment, stage. Each base library has the following name format:

  adminprfx. .&C1SY..&C1SU..&C1EN(1,3)&C1S#.*typename*

  typename

  Specifies a language version.

Edit this table as follows:

1. Add or remove element type rows to include the language types used at your site. The type names appear in the table Node column.
2. Do not change the type names for PROCESS, TABLE, DELTA, LISTLIB, LLISTLIB, LOADLIB or OBJLIB, because these have a specific meaning to CA Endevor SCM and must not be altered. For these types, unless you are changing the format of the DELTA, LISTLIB, or LLISTLIB libraries, limit your changes to the Sizeunit, Primary, Secondary and Directory.
3. (Optional.) If you want to define ELIB BDAM data sets for DELTA, LISTING, and LLISTING data sets, specify the following values:

   Note: The #OPTION member DELTA and LISTLIB data set format must be the same as the format specified in the T#ADMALL member. By default, the DELTA, LISTLIB and LLISTLIB are created as ELIB BDAM. You can leave them as is, or change them to PDS or PDSE by editing the table.

   Note: For more information about ELIB BDAM data sets, see the Administration Guide.

   - PrimPages—Specifies the number of primary pages for the ELIB initialization program to allocate. The value of this parameter is placed into the ALLOCATE PAGES parameter;

     ALLOCATE PAGES = (&PrimPages,&SecdryPages)

   - SecdryPages—Specifies the number of secondary pages for the ELIB initialization program to allocate.. The value of this parameter is placed into the ALLOCATE PAGES parameter;

     ALLOCATE PAGES = (&PrimPages,&SecdryPages)

   - ResrvePages—Specifies the number of pages for the ELIB initialization program to reserve. The value of this parameter is placed into the RESERVE PAGES parameter;

     RESERVE PAGES = &ResrvePages

- DirPages—Specifies the number of directory pages for the ELIB initialization program to allocate.. The value of this parameter is placed into the DIRECTORY PAGES parameter;

  ```
  DIRECTORY PAGES = &DirPages
  ```

Important: The same types must be listed in the T#ADMALL andT#ADMTYP tables. If you add or remove type rows from any of these tables, you must make a corresponding change to the other tables.

T#ADMSUB--SubSystem Names for ADMIN Environment

The T#ADMSUB table specifies the name of the subsystems for the ADMIN environment. The delivered table specifies the following:

- System: ADMIN.
- Subsystem: ENDEVOR.
- Description: CA Endevor SCM tables and processors.

Usually, no change is needed to this table.

Important: The system names must be the same in each of these tables: T#ADMSYS and T#ADMSUB. If you add or remove a system row from the T#ADMSYS table, you must add or remove the associated rows in the T#ADMSUB table.

T#ADMSYS--System Names for ADMIN Environment

This table specifies the ADMIN environment systems to be implemented. The delivered table specifies the following:

- System: ADMIN.
- Description: CA Endevor SCM administration.

The system name is used to build the following processor output libraries.

Important: The first system referenced in the table is used to build the processor output library names.

As delivered the following tables will be built:

adminprfx.ADMIN.PROCESS.ADM1.LOADLIB

adminprfx.ADMIN.PROCESS.ADM1.LISTLIB

adminprfx.ADMIN.PROCESS.ADM2.LOADLIB

adminprfx.ADMIN.PROCESS.ADM2.LISTLIB

system_name

Specifies the name of the system in the ADMIN environment.

Usually, no change is needed to this table.

Important: The system names must be the same in these tables: T#ADMSYS and T#ADMSUB. If you add or remove a system row from the T#ADMSYS table, you must add or remove the associated rows in the T#ADMSUB table.

T#ADMTYP--Types for the ADMIN Environment

This T#ADMTYP table specifies the type definitions to be created in the ADMIN environment.

Edit this table as follows:

1. Add or remove element type rows to include the language types used at your site.
2. Do not change the type names for PROCESS or TABLE, because CA Endevor SCM uses these to hold the CA Endevor SCM processors and configuration tables and these must be set to the values provided.

Important: The first processor group listed for each type is used as the default processor group for that type.

Important: The same types must be listed in the T#ADMALL andT#ADMTYP tables. If you add or remove type rows from any of these tables, you must make a corresponding change to the other tables.

T#ALLOC--Allocations for Non-ADMIN Environments

The T#ALLOC table allocates data sets for all environments except the ADMIN environment. The UNCLMED environment and the PRD environments apply an allocation factor specified in #OPTION to the values specified in this table.

This table enables the creation of the following data sets for each element type defined in the T#TYPES table.

- One DELTA, LISTLIB, LLISTLIB, LOADLIB, and OBJLIB library for each element type for the application or standard environment.
- One base library for each element type for each system, subsystem, environment, stage. Each base library has the following name format:

```
aprfx. .&C1SY..&C1SU..&C1EN(1,3)&C1S#.typename
```

typename

Specifies a language version.

Edit this table as follows:

1. Add or remove element type rows to include the language types used at your site. The type names appear in the table Node column.
2. Do not change the type names for PROCESS, TABLE, DELTA, LISTLIB, LLISTLIB, LOADLIB or OBJLIB, because these have a specific meaning to CA Endevor SCM and must not be altered. For these types, unless you are changing the format of the DELTA, LISTLIB, or LLISTLIB libraries, limit your changes to the Sizeunit, Primary, Secondary and Directory.
3. (Optional.) Specify the following values, if you want to define ELIB BDAM data sets for DELTA, LISTING, and LLISTING data sets: (ELIB BDAM is the default.)
   - PrimPages— Specifies the number of primary pages for the ELIB initialization program to allocate. The value of this parameter is placed into the ALLOCATE PAGES parameter;

     ```
     ALLOCATE PAGES = (&PrimPages,&SecdryPages)
     ```

   - SecdryPages— Specifies the number of secondary pages for the ELIB initialization program to allocate. The value of this parameter is placed into the ALLOCATE PAGES parameter;

     ```
     ALLOCATE PAGES = (&PrimPages,&SecdryPages)
     ```

   - ResrvePages— Specifies the number of pages for the ELIB initialization program to reserve. The value of this parameter is placed into the RESERVE PAGES parameter;

     ```
     RESERVE PAGES = &ResrvePages
     ```

   - DirPages— Specifies the number of directory pages for the ELIB initialization program to allocate.. The value of this parameter is placed into the DIRECTORY PAGES parameter;

```
DIRECTORY PAGES = &DirPages
```

Note: The #OPTION member affects the UNCLMED environment and the PRD environments as follows:

- The ProductionSizeFactor specified in the #OPTION member in the uprfx.uqual.BPILIB library is applied to the *production environment* allocations, by the #5ALLOC job, when building the allocation statements. For example, if you specify a Primary value of 50 for the ASMPGM base library and the ProductionSizeFactor value is 3, the data set is allocated with 150 primary cylinders. This factor is also applied to the Secondary and Directory columns when processing the production environment. The ProductionSizeFactor is skipped when allocating ELIB DELTAs or LISTINGs.
- The UnclaimedSizeFactor specified in the #OPTION member in the uprfx.uqual.BPILIB library is applied to the *unclaimed environment* allocations, by the #4UNCLM job, when building the allocation statements. For example, if you specify a Primary value of 40 for the ASMPGM base library and the UnclaimedSizeFactor value is 3, the data set is allocated with 120 primary cylinders. This factor is also applied to the Secondary and Directory columns when processing the unclaimed environment. The UnclaimedSizeFactor is skipped when allocating ELIB DELTAs or LISTINGs.

T#ENVMTS--Environment and Stage Names

The T#ENVMTS table specifies the environments and stages to be created.

As delivered the implementation creates the following environments and stages:

Envname	Stg1ID	Stg1nme	Stg2ID	Stg2nme	Entrystg#	NextEnv	NextStgID
ADMIN	1	ADMIN1	2	ADMIN2	1		
DEV	1	DEV1	2	DEV2	1	QAS	2
QAS	1	QAS1	2	QAS2	1	PRD	2
PRD	1	PRD1	2	PRD2	1		
UNCLMED	1	UNCLMED1	2	UNCLMED2	2		
ARCHIVE	1	ARCHIVE1	2	ARCHIVE2	2		
EMER	1	EMER1	2	EMER2	1	PRD	2

Edit this table as follows:

1. Add or remove environment rows to include the environments used at your site. All values in the data rows must be entered in upper case character mode. The first three characters of the environment name are used to build data set names. Therefore, the first three characters of the names must be unique across environments. If the environment names do not contain three characters, a pad character ($) is added to the environment name when allocating the data sets for that environment.
2. Do not change the environments names for ADMIN, PRD, UNCLMED, or ARCHIVE, unless you have business reason to do so. If you change these, you must also change the name in the O#ENVMTS REXX member, located in the uprfx.uqual.BPILIB data set. The purpose of this member is to identify which of the environments is the administrator, production, unclaimed element and archive element environments.
3. If you decide you do not need the UNCLMED environment, comment out the UNCLMED row in this table prior to executing the #1TABLES job. Job #1TABLES builds the C1DEFLTS table.

   Note: Also skip the execution of the #4UNCLM job.

T#GLBSEQ--Global Type Sequence Member

The T#GLBSEQ table contains the information required to create the global type sequence member, GLBLTYPE in the uprfx.uqual.PARMLIB. CA Endevor SCM uses this PARMLIB member to determine the order of type processing. This table is used in conjunction with the T#ADMTYP and T#TYPES tables. If data rows are added or removed from these tables, the T#GLBSEQ should also be reviewed to see if changes are required.

Only types that must be executed in a specific sequence should be included in this table. Types such as JCL and PROCS should not be included in this table. Following this rule will result in more efficient through put from the current action processing feature.

T#LIBRYS--Allocations for Site-Wide Data Sets

The T#LIBRYS table allocates CA Endevor SCM data sets that are used site-wide. The purpose of this model is to identify the fully qualified names of these data sets. The values in the Node column are used as the last qualifier of the fully qualified data set name as shown next:

- ACMROOT—ACM root and data sets.
- ACMXREF— Cross-reference data sets.
- ELMCATL—element catalog data sets.
- ELMCATL.EINDEX—element catalog index data sets.
- PACKAGE—Package file.
- PARMLIB—Parameter library.
- MCF—Master Control File for each environment-stage combination.

Do not add or remove any rows from this table, however, you should review the type of data set specified in the MODEL column and the data set size attributes.

It is highly recommended that you do not change the Node, MODEL or TBLOUT values in this table. DO NOT change the MCF name. If you have a business reason to change one of the other node names, you must also change the name in the O#LIBRYS REXX model, located in the iprfx.iqual.CSIQCLS0 data set.

T#SUBSYS--Subsystem Names for All Environments Except All Standard or Application Environments

The T#SUBSYS table names the Subsystems for each System and provides a description of each Subsystem for all environments except the ADMIN and UNCLMED environments.

Edit this table as follows:

1. Change the default to the Systems and Subsystems appropriate for your business model.
2. Add or remove corresponding rows in the T#SYSTMS table, if you changed the T#SUBSYS table in step 1.

Note: The default subsystem names match the sample application.

T#SYMBLS--Symbol Names for All Environments Except ADMIN and UNCLMED

The T#SYMBLS table contains all the symbol overrides to be defined for the standard or application environments.

This table contains the symbol overrides to be defined for the standard environments. You must specify the type, processor group, environment, stage, processor type, symbol, and override value for each row in the table. You can wildcard the processor group, environment, and Stg# columns. However, no partial wildcarding is supported for the environment or stage column.

The stage column if wildcarded, will create symbol definitions for stage 1 and stage 2.

The environment column if wildcarded, will create symbol definitions for all environments (DEV, QAS, PRD, EMER) except for ADMIN, UNCLMED, and ARCHIVE. They are excluded because the Administrator environment has symbol creation steps in #3ADMIN, and the Unclaimed and Archive environments set processors to *NOPROC*. However, you may define symbols for any of the environments by creating a unique row for that location in the table, placing the environment name directly in the Envname column. In other words, although the wildcarding feature excludes the Administrator environment, you can still create symbols for that environment using this table, by placing ADMIN in the Envname column.

All symbol definitions with wild carded environments will be created first, followed by specific symbol definitions, allowing for additional flexibility when establishing the symbol rows. There are many processor symbolic overrides being created, as delivered. You can delete or add additional rows to the symbol table as needed.

T#SYSTMS--System Names for Standard or Application Environments

The T#SYSTMS table contains all the environment systems to be implemented. This table comes preset with values from the Endevor Sample Application. All values in the data rows must be entered in upper case character mode except for the description, which can be entered in mixed mode.

This table is related to the T#SUBSYS table. If you add or remove a system row from the T#SYSTMS table, you must add or remove the associated rows in the T#SUBSYS table.

T#TYPES--Types for Non-ADMIN Environments

The T#TYPES table contains the types to be created within the DEV, QAS, PRD, EMER, ARCHIVE and UNCLMED environments.

Do not alter the values in the PROCESS rows. This type is used to hold the CA Endevor SCM processors and must be set to the values provided.

If type definition rows are added or removed from the T#TYPES table, the corresponding row must be added or removed from the T#ALLOC table. The T#GLBSEQ table should also be reviewed to determine if rows must be added or removed from that table.

Prepare the BPI Job Card

Edit the #JOBCARD member in the uprfx.uqual.BPILIB library so the information conforms to your site standards. Do not alter the title. The &TBLOUT symbol will resolve to the job name.

```
//BPI@ENDV JOB (#########),'BPI &TBLOUT ',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID
```

The job card is ready for use by the BPI jobs.

Note: When the #0SETUP job is executed, this job card is automatically placed at the beginning of each of the other BPI jobs.

Note: After the BPISTART job is executed, the remaining BPI process is executed from the BPILIB library.

Define Symbol Values for BPI Jobs

To define values for variables used by the BPI process to complete the configuration, you edit the #OPTION member. These values are used to resolve variables that are defined in BPI models and REXX members.

Edit the uprfx.uqual.BPILIB(#OPTION) member and answer all the questions. The questions are in the format of REXX comments and the answers are in the format of REXX statements. These statements are used as symbol values during the implementation process. For example, many of the C1DEFLTS table site parameter values are derived from the information specified in this member. The literals on the left side of the equal symbols are REXX variables and must remain in mixed character mode. The values you enter on the right side of the equal symbols are data values and must be entered in upper case character mode, unless otherwise stated; for example, comments, descriptions and notes can be entered in mixed character mode.

Submit BPI Jobs

You run the BPI as a set of jobs that use information in the input tables and the #OPTION member. The first job you submit is the #0SETUP job, which builds the other jobs that you must submit. All the jobs are located in the BPILIB library and must be run in the specified order. Some of these jobs build additional jobs. These additional jobs are submitted automatically, if the #OPTION member SubmitJob parameter is Y. Submit the jobs to complete the set up.

Follow these steps:

1. Submit the setup job, #0SETUP.

   Checks for the existence of data sets and builds and tailors additional jobs. This job builds nine additional jobs and saves them into the uprfx.uqual.BPILIB library. This job checks for the existence of the CA Endevor SCM site-wide data sets it is about to allocate. If any of the data sets already exist, the job terminates with a return code of 12 and a report is produced to indicate which of the data sets already exists. The purpose of this check is to ensure the data sets are not accidentally deleted.

   The #0SETUP job also builds additional BPI jobs. Jobs that are built by the setup job are also stored into the BPILIB library. The #JOBCARD you coded is copied to the beginning of each of these jobs.

2. Submit the #1TABLES job.

   This job builds, assembles and link-edits the C1DEFLTS and ESYMBOLS tables. No additional jobs are created by this job. This job also assembles and link-edits the ENCOPTBL and ENDICNFG configuration tables.

3. Submit the #2LIBRYS job.

   Allocates Endevor Site-wide Data sets. Builds the global type sequence member and saves it into the PARMLIB. This job builds one additional job.

4. Submit the #3ADMIN job.

   Creates the entire ADMIN Environment. Adds processors and configuration tables. This job builds four or five additional jobs, depending on the allocation specifications.

5. Submit the #4UNCLM job.

   Creates the entire UNCLMED Environment. This job builds four or five additional jobs, depending on allocation specifications.

6. Submit the #5ALLOC job.

   Allocates all Endevor Application Data sets. This job builds additional jobs depending on the number of system/subsystem combinations specified in the T#SUBSYS BPI table.

7. Submit the #6DEFSYS job.

   Defines Systems for the DEV, QAS, PRD, EMER and ARCHIVE Environments. This job builds one additional job.

8. Submit the #7DEFSUB job.

   Defines Subsystems for the DEV, QAS, PRD, EMER and ARCHIVE Environments. This job builds one additional job.

9. Submit the #8DEFTYP job.

   Defines Types for the DEV, QAS, PRD, EMER and ARCHIVE Environments. It will also define Processor Symbols for any of the environments specified. This job builds one additional job.

10. Submit the #9DEFGRP job.

    Defines processor groups for the DEV, QAS, PRD, EMER and ARCHIVE Environments. It will also define processor symbols for any of the environments specified.

11. Manually delete the *uprfx.uqual*.BPIWRK library after every iteration of the BPI process.The BPIWRK data set provides no value once the implementation is complete.

The Iterative BPI Process

We recommend that you execute the BPI process in iterations. For the first iteration, you might implement one System (or application). This lets you become familiar with how to edit the input tables and how the jobs work. After the first iteration, you could execute the BPI to implement an entire application (an application can consist of multiple Systems and Subsystems). Then you could do additional iterations for each additional application. For the first iteration, you perform all the steps. For subsequent iterations, you re-edit the BPI input tables and then rerun the jobs beginning with job #5ALLOC.

For the first iteration, do the following:

1. Submit BPISTART.
2. Complete all the BPI input tables.
3. Edit the #JOBCARD.
4. Edit #OPTION to enable symbolic substitution.
5. Execute the BPI jobs #0SETUP through #9DEFGRP, and submit the additional jobs that get created.

For the second and subsequent iterations, do the following:

1. Edit the values in the tables.
2. Execute BPI jobs #5ALLOC through #9DEFGRP, and submit the additional jobs that get created.
3. Repeat these steps as many times as needed to add all the Systems you want to put under the control of CA Endevor SCM at this time.