CA Endevor® Software Change Manager

Security 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 External Security Interface (CA Endevor External Security Interface)
- CA Top Secret® for z/OS (CA Top Secret)
- CA ACF2™ for z/OS (CA ACF2)

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

- The Action Initiation Security Control Point (see page 35), How to Define the User Security Table (see page 42), Name Equates Table (see page 60), How to Define SAF Authorization Levels (see page 64), Securing Actions Using the FUNCEQUE Entry (see page 67), How to Specify a Substring of a Keyword (see page 71), The Action Initiation Security Control Point (Extension) (see page 85), Define Security Rules for Action Initiations (see page 99), The Default Authorization Value (see page 86), and The Action Initiation Security Control Worksheet (see page 120)— Updated to add the Alter action.

Version 16.0

- USS Supported Files and the Alternate ID (see page 26)— Added to replace the obsolete topic Alternate ID Support for UNIX Files.

Version 15.0

- The Alternate ID (see page 16)— Updated to specify that the alternate ID cannot control any data set under the control of the CA Common Services component CA L-Serv.
- Using the Alternate ID with REXX and the Internal Reader (see page 25)— Updated regarding submitting jobs to an internal reader from a REXX exec when the internal reader is allocated in an INTRDR DD statement in the processor.
- How to Specify a Substring of a Keyword (see page 71)— Updated to add EABKO and EABKI values for PKGSUBFC.
- The Package Actions Security Control Point (see page 88)— Updated to add EABKO and EABKI values for PKGSUBFC.

Contents

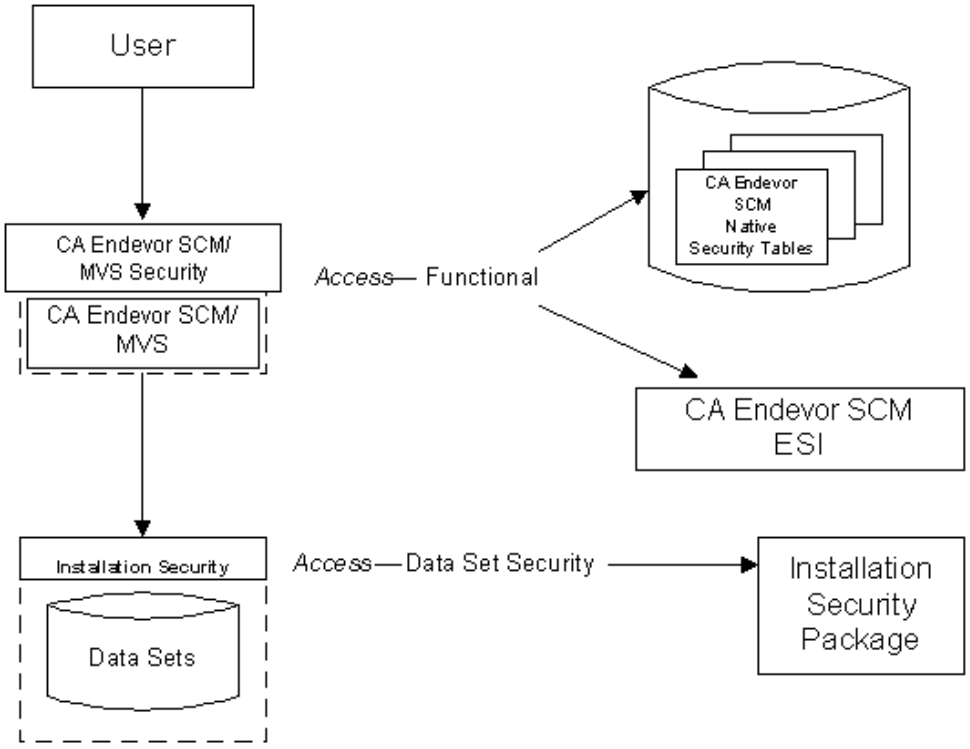
Chapter 1: Introduction

This section contains the following topics:

Security Options

To provide a comprehensive security program for CA Endevor SCM Change Manager, you must address security issues in two essential areas: data set security and functional security.

The following figure depicts the relationship between CA Endevor SCM and functional and data set security.



As previously illustrated, CA Endevor SCM uses one of two available means of providing functional security: native security tables or CA Endevor External Security Interface (ESI).

Data Set Security Types

CA Endevor SCM does *not* provide data set security. Data set security is performed by a site security package, such as:

- RACF
- CA ACF2 for z/OS
- CA Top Secret for z/OS

Data set security involves preventing unauthorized access to the data sets that CA Endevor SCM uses. Two approaches are available for controlling access to your physical data sets:

Program path protection

Gives the CA Endevor SCM system access to the data sets it maintains. You must go through CA Endevor SCM to perform maintenance on these data sets.

Standard data set security

Gives users direct access to data sets maintained by CA Endevor SCM. Although unauthorized access to data sets is prevented, authorized users can maintain these data sets without going through CA Endevor SCM.

Implementing data set security in addition to functional security is recommended to control access to data sets by CA Endevor SCM users.

Note: For more information, see Implementing Data Set Security (see page 15).

Functional Security

Functional security involves protecting CA Endevor SCM inventory functions from unauthorized access. These functions include access to menu options, the ability to perform certain actions against certain inventory areas, and other secured CA Endevor SCM options.

Functional security is provided by CA Endevor SCM, unlike data set security. You must choose between one of two methods for providing functional security:

Native Security Tables

Control environment access, primary and foreground menu options, and action authorization.

External Security Interface (ESI)

Controls environment access, primary and foreground menu options, action authorization, package actions, and concurrent action processing authorization, as well as allowing you to store security rules under your site security package. In addition, ESI allows you to customize your functional security capabilities.

Security Control Points

During processing, CA Endevor SCM performs security checks to allow or deny a user access to certain inventory areas and inventory actions. These checkpoints are referred to as *security control points.*

The security control points listed determine the appropriate level of access to system inventories and functions:

Environment Selection

Verifies your access to a requested environment.

Primary Options

Verifies your access to particular operations appearing in the Primary Options menu.

Foreground Options

Verifies your access to actions available on the Foreground Options menu.

Action Initiation

Verifies your access to actions such as DISPLAY, ADD/UPDATE, RETRIEVE, or GENERATE.

Package Actions

Verifies a user's access to package actions such as CREATE, CAST, DYNAMIC, REVIEW, and EXECUTE (applies to ESI only).

Concurrent Action Processing

Verifies a user's access to request concurrent action processing.

When security control points are reached, CA Endevor SCM checks access privileges defined in one of the following security configurations:

- Native security tables
- ESI, interfacing with an external security product such as RACF, CA ACF2 for z/OS, or CA Top Secret for z/OS

For example, when CA Endevor SCM reaches a security control point, the system reviews the native security tables and determines whether a user is allowed to perform certain inventory actions against a portion of the inventory. CA Endevor SCM then hides inventory elements the user is not permitted to access and functions the user is not allowed to perform. In this way, ESI allows you to customize your functional security capabilities.

Native Security

Native security uses the following three security tables to record security rules for access to inventory levels and functions:

Access Security Table (one for an installation)

Defines the environments to which you have access.

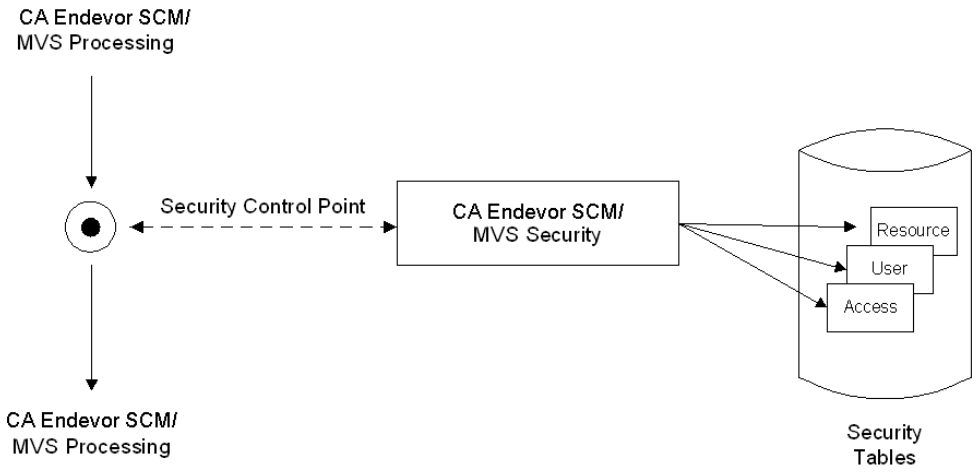
User Security Table (one for an environment)

Defines the menu options available to you after access to an environment is obtained. Further, this table defines actions allowed within the environment, by user, for each system and subsystem.

Resource Security Table (one for an environment)

Enforces naming conventions at the system/subsystem and element level.

The following figure shows how control points can control access to CA Endevor SCM processing functions by checking native security tables.

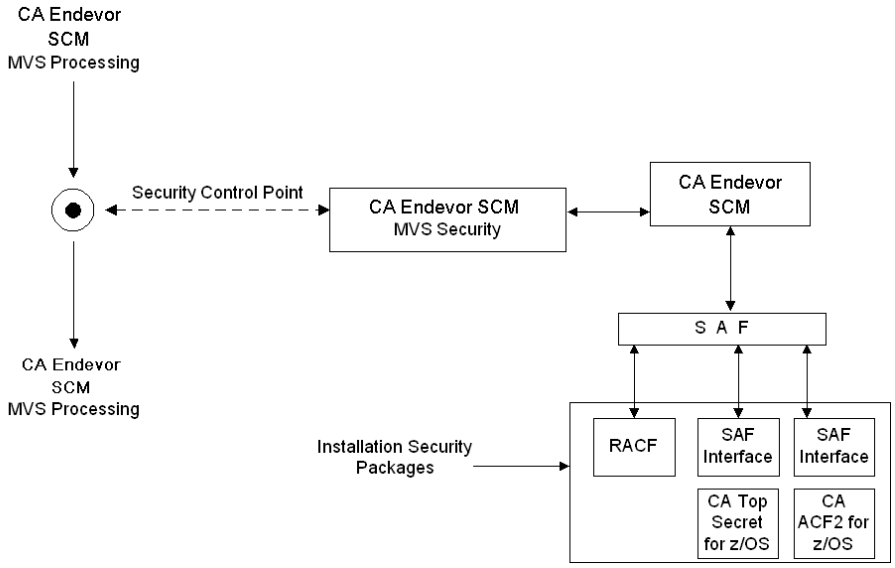


As previously illustrated, upon entry into the system, your access to CA Endevor SCM functions is controlled by CA Endevor SCM Security Tables or the CA Endevor SCM (ESI). Site security provides protection of system data sets.

The External Security Interface (ESI)

ESI is an optional feature that allows you to secure CA Endevor SCM's access and action functions through IBM's System Authorization Facility (SAF), using the site security package on your system. It does this by allowing you to define the rules for functional security in your site security package (RACF, CA ACF2 for z/OS, CA Top Secret for z/OS) rather than in the native tables supplied with CA Endevor SCM. For more information on how to enable and use CA Endevor SCM ESI, see Enabling External Security Interface (ESI) (see page 55).

The following diagram shows how ESI interacts with site security packages.



If ESI is enabled, security rules must be defined to the site security package. In the previous diagram, CA Endevor SCM uses IBM's System Authorization Facility (SAF) calls to query the installed security package instead of using native security tables.

Selecting Your Security Option

To protect CA Endevor SCM and its data sets, you need to install a security function that prevents unauthorized access to your system. You may need to choose from among a number of data set and functional security options.

Data Set Security Methods

Options for data set security include the following:

Program path protection

- Allows authorized users access to a data set through an authorized program.
- Permits library updates only through CA Endevor SCM.
- Provides secure preventative control.

Standard data set protection

- Allows authorized users direct access to data sets.
- Permits library updates outside of CA Endevor SCM.
- Provides a measure of preventative control.

No data set protection

- Permits unlimited access to data sets.

  Note: You can use CA Endevor SCM Footprint Exception Reporting to detect unauthorized updates. For more information on reporting, see the *Reports Guide*.

Functional Security Methods

Functional security can be provided by implementing one of the two following CA Endevor SCM security methods.

Native security tables

- Provides basic functional security.
- Allows access to environments.
- Permits primary and foreground menu options.
- Requires authorization of actions.

ESI

- Provides basic functional security.
- Allows extension of functional security through customization.
- Integrates with existing security package, such as RACF, CA ACF2 for z/OS, or CA Top Secret for z/OS.

Implementing Security

We recommend a comprehensive approach to system security that ensures functional and data set security. A carefully planned security program ensures the proper levels of access and data set security.

You should address security during the final testing of your first CA Endevor SCM application. The initial setup and testing steps for implementing a new application should not be disrupted by overly restrictive security rules.

How to Enable Data Set Security

Follow this process to implement data set access security and ensure your success.

1. Lay out your data set access requirements in a simple, non-technical form for review.
2. Build your data set security profiles in Warning Mode.
3. Test your security implementation and monitor any warnings you receive.
4. Set data set security profiles to Live Mode.
5. Monitor security violations on an ongoing basis.

How to Enable Functional Security

Decide which means of functional security you want to use: native security tables or ESI. You can only use one method.

How to Implement Native Security

Follow this process to implement functional security using native security tables and ensure your success.

1. Plan security for a pilot application.
2. Define the three native security tables.
3. Access Security Table.
4. User Security Table.
5. Resource Security Table.
6. Activate the security tables.
7. Test your security implementation, monitor violations, and correct tables.

How to Implement ESI Security

Follow this process to implement ESI security and ensure your success.

1. Plan security for a pilot application.
2. Customize the ESI Security Definition Table, BC1TNEQU.
3. Lay out ESI security profiles.
4. Build ESI security profiles in Warning Mode.

5. Customize the C1DEFLTS Table

   ESI is activated.

6. Test ESI security and monitor warnings.
7. Set ESI security profiles to Live Mode and monitor violations.

SCL Statement Syntax Convention

CA Endevor SCM uses the IBM standard for representing syntax.

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

Chapter 2: Implementing Data Set Security

This section contains the following topics:

Data Set Security

Data set security refers to the protection of files accessed and maintained by CA Endevor SCM. The alternate ID feature allows you to restrict access to specified data sets by the user ID specified in the customer default table. It automatically determines which data sets are controlled by CA Endevor SCM. This practice allows CA Endevor SCM to perform updates only on the set of libraries you specify.

Note: You must add the names of non-CA Endevor SCM programs to the Authorized Program Table (C1GTAPGM), if these programs are invoked within a processor and need to run APF-authorized. For more information, see Authorized Program Table in the chapter "Logical and Physical Structure" in the *Administration Guide.*

Alternate ID Support

Some versions of program pathing do not support the ability to recognize top level programs through conventional means. A special interface has been developed to be used in conjunction with CA ACF2 for z/OS, CA Top Secret for z/OS, and RACF to enable you to perform data set security using alternate ID support. This interface allows manipulation of data sets and their contents through CA Endevor SCM, using an alternate user ID.

This method of data set security ensures that updates to controlled data sets are performed only through CA Endevor SCM. External access to these files by CA Endevor SCM users is prohibited. For example, if an element is changed outside of CA Endevor SCM, then the CA Endevor SCM footprint is compromised.

Note: Alternate ID support for USS files and directories is subject to certain limitations. For more information, see Alternate ID Support for UNIX Files. (see page 27)

Important! We recommend using alternate ID support for data set security.

The Alternate ID

When defining your environment, you can optionally specify a RACF ID, known as the CA Endevor SCM Alternate ID, in your C1DELFTS to be used when accessing CA Endevor SCM controlled resources such as the MCF, package data sets, processor data sets, or the element catalog. If an Alternate ID has been specified it will be used instead of the user ID when accessing these resources.

If the OPEN SVC screen detects that the OPEN was issued from CA Endevor SCM I/O modules, and the target file appears to be an CA Endevor SCM file, then the TCBSENV field is swapped to the Alternate ID, causing the task to run under the security context of the Alternate ID. When the OPEN completes, the field is swapped back to its earlier value. This logic is bypassed if ALTID=N is coded.

Note: The External Security Interface (ESI) never switches to the CA Endevor SCM Alternate ID. ESI is used to determine whether a user has the right to perform certain CA Endevor SCM actions. It performs this check by issuing an RACROUTE REQUEST=AUTH call to validate a user's access.

What the Alternate ID Controls

When activated, the alternate user ID is used when any of the following data set categories are accessed. An access level of UPDATE is required for these data sets. However, for the Master Control File, Element Catalog File, Element Index File, Package Control File, and ELIB VSAM data sets, an access level of CONTROL is required.

- Master Control File
- Package Control File
- Element Catalog File
- Element Index File
- ACMQ root and XREF data sets
- Base/delta libraries*
- Processor load libraries
- Source input libraries (for example, include libraries)*
- Source output libraries (for example, load libraries)*
- All data sets contained in generate, move, or delete processors*
- CCID validation data set
- Package ship staging data sets

- Batch request data sets
- ISPF (foreground) data sets

*If these are USS files, alternate ID support is not available for delta libraries, and support is limited for USS files accessed in processors and user exits. For more information, see Alternate ID Support for UNIX Files. (see page 27)

What the Alternate ID Does Not Control

Several categories of data sets can only be accessed by the user's originating TSO user ID. These data sets include:

- Foreground ISPF Data Sets (for example, uid.C1TEMPRnMSGS, uidC1TEMPncntl, uid.C1#NTMPL.LIST)
- ADD/UPDATE FROM: *data set*
- RETRIEVE TO: *data set*
- ARCHIVE/TRANSFER TO: *data set*
- COPY FROM: *data set*
- RESTORE FROM: *data set*
- LIST FROM: *data set*
- TRANSFER FROM: *data set*
- PRINT TO: *data set*
- Batch request data sets
- Any data sets controlled by the CA Common Services component CA L-Serv
- JES2 files SYSIN and SYSOUT data sets

Note: The alternate ID is not swapped on the creation or deletion of a data set or PDS in a processor. The user's TSO user ID must have authority to create or delete the data set or PDS.

How to Activate the Alternate ID for Data Set Protection

To implement the Alternate ID support feature, you must specify the alternate user ID in the Defaults table, build profiles to protect the libraries, and grant update authority to the Alternate ID for those libraries as indicated in the following steps. An additional step is required to enable Alternate ID protection for USS files and directories.

Note: Alternate ID support for USS files and directories is subject to certain limitations. For more information, see USS Supported Files and the Alternate ID (see page 26).

1. Update the TYPE=MAIN macro in the CA Endevor SCM Defaults table (C1DEFLTS) to include (or update) the RACFUID parameter. Set the value for the *userid* variable to the The RACF, CA ACF2, or CA Top Secret user ID that is to be used for data set authorization checking.

   RACFUID =*userid*,

   Indicates the user ID required for Alternate ID support.

2. Assemble and link C1DEFLTS into the authorized load library and issue an LLA refresh for the library.

3. Use your site's security system to build profiles to protect the libraries, and grant update authority to the Alternate ID for those libraries.

4. In order to perform periodic maintenance against the various files under CA Endevor SCM control such as PDS compress and VSAM Repro, you must grant your administrator the same permissions as the RACFUID. Also, RACFUID needs READ access to the CONLIB. Some types of file maintenance, such as full file restore, may require a higher level of authority than that granted to the RACFUID. Be sure to allow someone the appropriate authority to address this event.

5. If you want to enable Alternate ID protection for USS files and directories, turn on the following option in the Options table:

   ENABLE_ALTID_USS_SECURITY=(ON,*nn*)

   Indicates whether Alternate ID support is enabled for UNIX USS files.

   ON

   The UNIX files managed by CA Endevor SCM will be owned by the Alternate ID. The file owner (alternate ID) will have read, write, and execute access to new base and source output files. This support is limited to UNIX base libraries, include libraries, and source output files that are defined as UNIX directories. Some support is also provided for UNIX files accessed in processors and user exits.

   OFF

   The UNIX files managed by CA Endevor SCM will **not** be owned by the Alternate ID. This is the default.

   nn

The UNIX file permission bits (rwx) for the owning group and other users, given to new and base source output files. The default is 55, meaning that the owning group and others will have both read and execute access.

Example: C1DEFLTS Parameter RACFUID

This example shows how to activate the Alternate ID for data set protection for your site.

```
C1DEFLTS TYPE=MAIN,                                              X
               ACCSTBL=,              ACCESS SECURITY TABLE NAME   X
               ACMROOT=,              ACM INDEX ROOT DATA SET NAME X
               ACMXREF=,              ACM INDEX XREF DATA SET NAME X
                 .
                 .
                 .
               RACFUID=userid,          Alternate ID USERID        X
```

Enabling the Alternate ID for UNIX Files

To use alternate ID security for UNIX files, this feature must be enabled by the administrator as follows:

- Specify an alternate ID value in the RACFUID field of C1DEFLTS. For more information, see How to Activate the Alternate ID for Data Set Protection. (see page 18)
- Set the ENABLE_ALTID_USS_SECURITY option to ON in the Optional Features table. The default for this option is OFF. For more information, see How to Activate the Alternate ID for Data Set Protection. (see page 18)
- Enhance the security definition of the alternate ID to include UNIX security. Consult with your site's security administrator and see the appropriate documentation for the External Security Manager (CA-TSS, CA-Top Secret, or RACF).
- If you have existing UNIX files and directories that you want to protect, ensure that the permissions are set as shown next:

Security Profile	Permissions		
	Read	Write	Execute
Owner (alternate ID)	X	X	X
Group	X		X
Other	X		X

To set the permissions on UNIX files and directories, follow these steps:

1. Use the 'chown' (change owner) UNIX command to change the owner to the alternate ID and the group ID to a group that includes the alternate ID. (The 'chown' command must be issued by a superuser.)
2. Use the 'chmod' (change permissions) command to change the permissions of directories and files to those shown in the previous table.
3. If you want to restrict read access to the directories to the alternate ID, omit the read permission for group and other.

How the Alternate ID Works with Processors

If the alternate ID is activated for data set protection at your site, you can specify whether the alternate ID is used in processors. Processor security is controlled by the EXEC statement ALTID parameter. Code one of the following options.

ALTID=Y

The default. Swaps the security context to that of the alternate ID for the duration of the processor step.The data sets specified in a processor step are allocated under the user ID. When the processor program is run, it runs under the alternate ID, so that access to the data sets (with the exception of JES2 files) occurs under the alternate ID.

ALTID=N

If you do not want the processor step to run under the security context of the alternate ID, then code ALTID=N on the exec statement for that step. Security context remains that of the user ID for the duration of the processor step. If your processor step includes a file whose output destination is the internal reader, the job step will run under the user ID.

Security for internal reader jobs is effected by the Optional Features Table (ENCOPTBL) option INTRDR_ALTID and processor ALTID settings as shown in the following table.

Processor ALTID= Setting	ENCOPTBL INTRDR_ALTID= Setting	Effects
Y or blank (default)	OFF (default)	Any processor step that includes an INTRDR DD card will be run under the user ID. No swapping to the altid for data access will occur. The job submitted to the internal reader will run under the user ID.
N	OFF (default)	Processor step runs under the user ID, therefore internal reader job will be submitted under user ID.
Y or blank (default)	ON	Processor step runs under the altid, therefore internal reader job will be submitted under altid.

N	ON	Processor step runs under the user ID, therefore internal reader job will run under the user ID. Note: ALTID=N overrides INTRDR_ALTID=ON.

Note: If the job is being submitted by a REXX exec, then the LGNT$$$I/O logic must be used in the REXX, regardless of whether the internal reader is allocated in the REXX itself or with an INTRDR DD statement in the processor. For more information, see Using the Alternate ID with REXX and the Internal Reader (see page 25).

Note: JES2 files are always opened under the user ID, regardless of the setting of ALTID in the processor step.

More information:

How Security Checking Works with JES2 Data Sets (see page 22)

How the Alternate ID Works for User Exits

If the alternate ID is activated for data set protection at your site, you can specify whether the alternate ID is used in user and package exits. Security for user and package exits is controlled by the user exit table C1UEXITS parameter USE_ALTID=. You can specify one of the following options:

USE_ALTID=Y

The default. Swaps to the alternate ID for data set security validation at OPEN time. Swaps security back to user ID when the data set open completes. Therefore, access to data sets is under the alternate ID, but the security context for all other processing is the user ID.

USE_ALTID=+

The security context is that of the alternate ID for the duration of the exit. Therefore, access to data sets occurs under the alternate ID as does all other processing. Use this option if you want to submit your internal reader jobs under the alternate ID.

USE_ALTID=N

The user exit runs completely under the security context of the user ID. The security context is never swapped to that of the alternate ID. All data set access and processing occurs under the user ID.

Note: For more information about the C1UEXITS table, see the *Exits Guide*.

Example: User Exit Security Checks Using Alternate ID for Data Set Access Only

This example shows how to code the user exit table to have data set access occur under the alternate ID, but all other processing occur under the user ID, for the duration of exit 1

```
@C1UEXIT EXIT#=1,NAME=C1UEXIT01,ANCHID=0,AUTH=YES,USE_ALTID=Y
```

Example: User Exit Security Checks Using Alternate ID for All Processing

This example shows how to code the user exit table to have data set access and all other processing occur under that alternate ID, for the duration of exit 2.

```
@C1UEXIT EXIT#=2,NAME=C1UEXIT01,ANCHID=0,AUTH=YES,USE_ALTID=+
```

Example: User Exit Security Checks Using User ID Only

This example shows how to code the user exit table to have data set access and all other processing occur under that user ID, for the duration of exit 3.

```
@C1UEXIT EXIT#=3,NAME=C1UEXIT01,ANCHID=0,AUTH=YES,USE_ALTID=N
```

Note: JES2 files are always opened under the user ID, regardless of the setting of the USE_ALTID in the user exit table.

How Security Checking Works with JES2 Data Sets

JES2 files are always opened under the user ID, regardless of the setting of ALTID in the processor step, or USE_ALTID in the user exit table. When a JES2 file opens, the security context reverts back to the user ID. When the open completes, the open screen resets the security context back to the Alternate ID. The JES2 data sets are accessed under the user ID to avoid the situation where an abend occurs and the processor dump information is written under the Alternate ID. This would cause security issues because the SYSUDUMP is most likely allocated in the main job (not the processor) and is therefore might not be accessible to the Alternate ID.

Using LGNT$$$I, LGNT$$$O Logic

When LGNT$$$I, LGNT$$$O logic is coded in a CLIST or REXX exec defined in a processor, the swap to the Alternate ID is done for the entire address space. This logic can be used to allow DB2 BINDS to run under the Alternate ID or to allow a job written to the INTRDR from the CLIST or REXX exec to run under the Alternate ID.

- When a LGNT$$$I file is opened during processor step execution, then the security context for the job step becomes that of the Alternate ID. Both the ASXBSENV and ASXBUSER values are set to those of the Alternate ID.
- When a LGNT$$$O file is opened during processor step execution, then the security context for the job step is restored to that of the user ID. Both the ASXBSENV and ASXBUSER values are set to those of the user ID. To successfully swap the ASXBUSER and ASXBSENV fields, opening LGNT$$$I and LGNT$$$O must occur in the same processor step as the DSN sub-command if a DB2 BIND is being executed, or as the write to the INTRDR if a job is being submitted to the internal reader. When the processor step ends, to account for the possibility that the LGNT$$$O logic was omitted, these values are always reset to that of the user ID. Additionally, if the processor step abends, the values are reset back to the user ID.

If ALTID=N is coded on the processor EXEC statement, then LGNT$$$I and LGNT$$$$O processing is bypassed and no security context swapping occurs.

Note: To successfully swap the ASXBUSER and ASXBSENV fields, opening LGNT$$$I and LGNT$$$O must occur in the same processor step as the DSN subcommand.

More information:

Using the Alternate ID with DB2 (see page 24)
Using the Alternate ID with REXX and the Internal Reader (see page 25)

Using the Alternate ID with DB2

Implementing the Alternate ID for DB2 allows binds of DB2 PLANS to occur under control of the Alternate ID, rather than the user's ID when the bind occurs during processor execution.

DB2 security may or may not be under RACF control. If it is not, then DB2 uses the ASXUSER value to determine the security context. If RACF is being used, then the value of the ASXBSENV field is used. The LGN$$$I/LGNT$$$0 mechanism updates both these fields.

ASXBUSER

The primary DB2 authorization ID and is a seven character field. The Alternate ID must also be seven characters or less.

ASXBSENV

The address of the ACEE control block that controls security of the address space level.

ASXBUSER is not used by RACF to determine the user ID authority for security groups. The Alternate ID must be assigned the DB2 authority necessary to issue the bind. Attempts to assign the Alternate ID to a group that has the necessary authority fails with DB2 authorization errors.

After opening file LGNT$$$I, both the AXSBUSER and ASXBSENV fields are updated to Alternate ID values. After opening LGNT$$$O, these values are reset to their original values. Opening these files triggers the Alternate ID swap. To successfully swap the ASXBUSER and ASXBSENV fields, opening LGNT$$$I and LGNT$$$O must occur in the same processor step as the DSN subcommand.

Example: CLIST invoked by BC1PTMP0 to implement the DB2 Alternate ID

```
WRITE ** SWAP ID TO ALTERNATE
ALLOC FILE (LGNT$$$I) DUMMY
OPENFILE LGNT$$$I
CLOSFILE LGNT$$$I
FREE FILE(LGNT$$$I)
WRITE ** BIND PLAN USING ALTERNATE ID
DSN SYSTEM(DSN6)
BIND PLAN(SQLASM01) MEM(SQLASM01) VAL(RUN) ACT(REP) ISO(RR) END
WRITE ** OPEN AND SWAP ID BACK TO USERS ORIGINAL ID
ALLOC FILE(LGNT$$$O) DUMMY
OPENFILE LGNT$$$O
CLOSFILE LGNT$$$O
FREE FILE(LGNT$$$O)
```

Using the Alternate ID with REXX and the Internal Reader

In a processor if you are submitting jobs to an internal reader from a REXX exec, to swap to the alternate ID, you must code the LGNT$$$I/LGNT$$$O logic into your REXX. This requirement applies whether the internal reader is allocated in the REXX exec or in an INTRDR DD statement in the processor.

Example: REXX, the Internal Reader, and the Alternate ID

This example shows how to code a REXX to swap to the Alternate ID for the internal reader.

```
/* REXX */
      "ALLOC F(JOBOUT1) SYSOUT(A) WRITER(INTRDR)"
      queue "SWAP TO ALTERNATE ID"
      Queue
      "ALLOC FILE(LGNT$$$I) DUMMY"
      "execio "queued()" diskw LGNT$$$I (finis"
      "FREE FILE(LGNT$$$I)"
      queue "//&PMFKEYA JOB (108200000),'KEIR',"
      queue "//             CLASS=A,MSGCLASS=X,"
      queue "//             NOTIFY=&PMFKEY"
      queue "//ENDEVOR  EXEC PGM=IEBGENER"
      queue "//SYSPRINT  DD DUMMY"
      queue "//SYSIN     DD DUMMY"
      queue "//SYSUT1    DD *"
      queue "TEST"
      queue "//SYSUT2    DD SYSOUT=*"
      Queue
      "execio "queued()" diskw jobout1 (finis"
      queue "SWAP BACK TO USERID"
      Queue
      "ALLOC FILE(LGNT$$$O) DUMMY"
      "execio "queued()" diskw LGNT$$$O (finis"
      "FREE FILE(LGNT$$$O)"
      "FREE FILE(JOBOUT1)"
      PUSH END
```

USS Supported Files and the Alternate ID

When using a USS base library, CA Endevor SCM source management accesses the library using the alternate ID. CA Endevor SCM also uses the alternate ID when it accesses a USS source output library in the CA Endevor SCM reserved processors BASICGEN and BASICDEL.

The security context of a processor step is determined by the following factors:

- **ALTID keyword—**The security context used for USS file access in processor steps is determined by the ALTID keyword. Processors can run under either the credentials of the user, or under the Alternate ID, so that USS outputs (created, copied, compiled, and so on) will have the Alternate ID as the owner or group owner. For more information about Alternate ID support, see How to Activate the Alternate ID for Data Set Protection, in the *Security Guide*.
- **PATHOPTS=(OCREAT)—**Any USS data sets created in processor steps using PATHOPTS=(OCREAT) are created using the security context of the user ID, prior to the invocation of the processor program. When used in the processor step, these data sets are opened using the security context of the processor step.
- **EXEC PGM=BPXBATCH—**The security context of a processor step executing the IBM USS utility program BPXBATCH depends on the following factors:
  - Whether BPXBATCH runs a shell script (PARM='SH') or directly executes an executable file (PARM='PGM'). *We recommend BPXBATCH be invoked directly as processor step.*
  - The settings of the Environment variables _BPX_BATCH_SPAWN and _BPX_SHAREAS.

  The following table shows combinations of these parameters and settings with the resulting security context of the processor step executing BPXBATCH.

BPXBATCH parameter	_BPX_BATCH_SPAWN	_BPX_SHAREAS	Security context
SH	ANY	ANY	Alternate ID
PGM	NO	n/a	Alternate ID
PGM	YES	NO	Alternate ID
PGM	YES	YES	User ID

- **BPXBATSL, BPXBATA2 and BPXBATA8—**These programs always run under the context of the user ID, because they process requests in the same manner as the last row in the prior table (_BPX_BATCH_SPAWN=YES, and _BPX_SHAREAS=YES).
- **BPXBATCH in an IKJEFT01 processor step—**

  Note: We do not recommend the use of BPXBATCH in an IKJEFT01 step, because the results can be unpredictable.

  Whether BPXBATCH invoked by a CLIST or REXX in a processor step executing IKJEFT01 executes under the context of the alternate ID depends on the following two *additional* factors:

  - Whether or not a LGNT$$$I swap was performed prior to the invocation of BPXBATCH.
  - Whether or not a prior USS service call was made in the job step prior to the BPXPBATCH call. For example, the prior USS service call could occur when a USS Base or Source Output Library was used during the CA Endevor SCM job step, or when a shell script was executed in this or any prior CA Endevor SCM action.

  The following table shows the security context results from these two additional factors when calling BPXBATCH from a processor step that executes IKJEFT01:

LGNT$$$I swap performed	Prior USS Service call made	Security context
Yes	No	Alternate ID
Yes	Yes	Failure
No	No	User ID
No	Yes	User ID

Note: To activate alternate ID support for data set protection, the administrator must perform certain steps. An additional step is required to activate alternate ID support for UNIX USS files. For more information, see How to Activate the Alternate ID for Data Set Protection (see page 18).

Note: CA Endevor SCM supports access to UNIX files and directories in certain situations. For more information, see USS Files and Directories in the appendix "Long Names and USS Files" in the *Administration Guide*.

Alternate ID Processing at Initialization

If alternate ID support is enabled, then in addition to changing the file permissions set for UNIX files managed by CA Endevor SCM, enablement of this option causes CA Endevor SCM to do the following processing at initialization.

- Attempt to establish the alternate ID UNIX security context by issuing an initUSP to acquire the USP (UNIX Security Packet) for the alternate ID. (If the alternate ID UNIX security context has already been established, this step is bypassed.)

- Dub the address space by issuing a getUID call, if the address space has not already been dubbed. (The getUID call simply returns the UID associated with the process. However, because it is a UNIX callable service, it forces the address space to be dubbed if it is not already dubbed.)

If any of the above fail, an error message is **not** issued until the first attempt to perform UNIX I/O is made. This done to avoid issuing these messages when no UNIX activity is taking place.

Program Pathing

Program pathing currently is available through both the CA ACF2 for z/OS and CA Top Secret for z/OS security packages. Data set security is implemented through a special Alternate ID interface, using an assigned RACF user ID and optional password for CA Endevor SCM.

When using either CA ACF2 for z/OS or CA Top Secret for z/OS with program pathing, you must define the top level programs that recognize CA Endevor SCM as the program in control for both foreground and batch processing. Resource rules *must* be written for the top-level programs. Depending on site requirements, resource rules may be written for other CA Endevor SCM programs. A list of the CA Endevor SCM's top-level programs requiring resource rules follows:

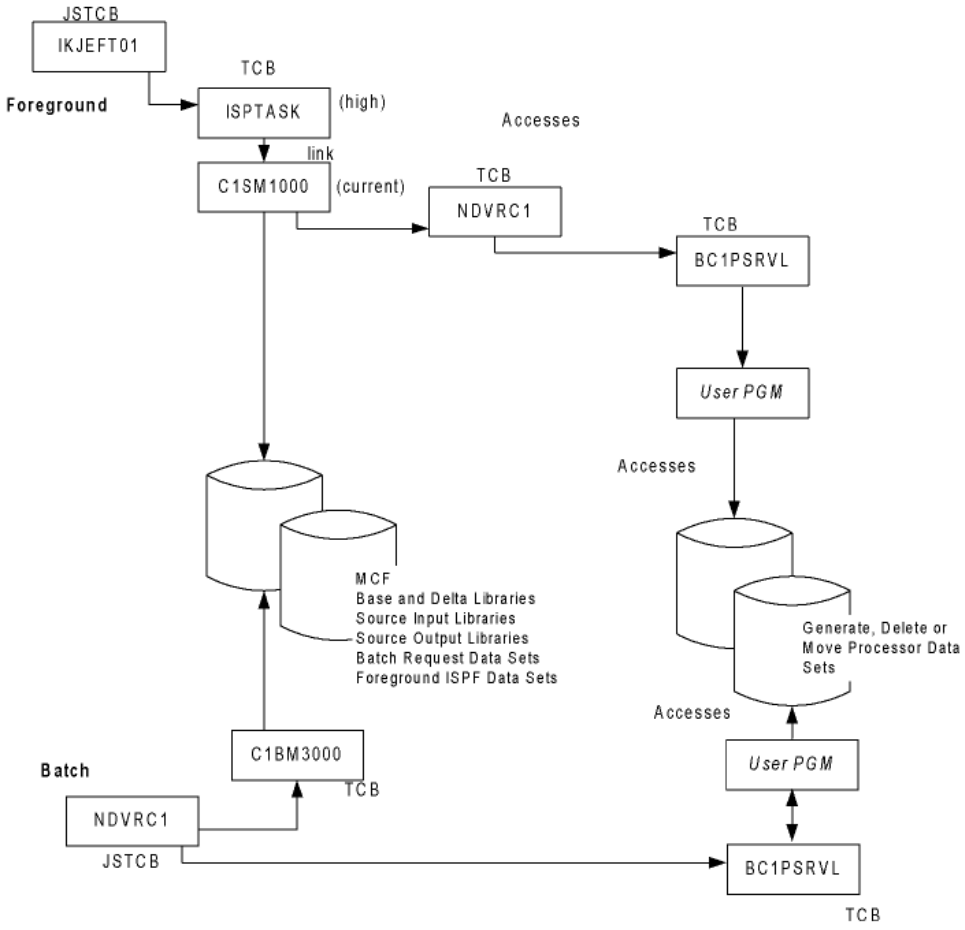
NDVRC1	C1BM5000	BC1PNLST	ENBE1000
BC1PSRVL	C1BM6000	BC1PNCPY	ENBP1000
C1SM1000	C1BR1000	BC1POPEN	IEFIIC
C1BM3000	BC1PNLIB	IEBCOPY	ENDIE000

Note: This list may change. Your site security needs may require resource rules for other CA Endevor SCM programs, and you can simply add these programs to the list above.

If you use CA Top Secret for z/OS, you can simplify the coding process by using the PRIVPGM option when writing a resource rule.

Data set security for CA ACF2 for z/OS and CA Top Secret for z/OS requires optional APARS and USERMODS from the appropriate CA Technical Support group.

The following diagram illustrates that CA Endevor SCM runs as an ISPF dialog under program ISPTASK.



As previously illustrated, ISPTASK must be addressed from a global point across the entire TSO environment, prior to attempting program pathing for CA Endevor SCM or any other ISPF dialog package.

Chapter 3: Enabling Native Security

This section contains the following topics:

Native Security Tables

Native Security Tables allow you to protect the functional side of your system. Native security protects your environments, systems, and subsystems from unauthorized access. In addition, element names can be defined for use only within specific systems or subsystems.

You can use the CA Endevor External Security Interface (ESI) instead of native security if you want to integrate your existing security package with CA Endevor SCM.

Note: For more information about ESI, see The External Security Interface (ESI) (see page 10). For more information about deciding whether ESI is an appropriate security option for your installation, see Security Options (see page 7).

CA Endevor SCM uses three tables to restrict access to functions and inventory levels:

Access Security Table

Defines the environment(s) to which each user has access. There is one Access Security Table for each installed site.

The Access Security Table must always be present in each site.

User Security Table

Defines the systems/subsystems available to each user within a particular environment, and for each system/subsystem combination, the level of activity for which each user is permitted access (browse-only, delete, add, and so forth).

There is one User Security Table for each environment.

Resource Security Table

Defines any elements that are restricted to particular a system/subsystem, and for each restricted element, the type of action for which the restriction applies.

This table is defined separately for each environment and there is only one Resource Security Table for each environment.

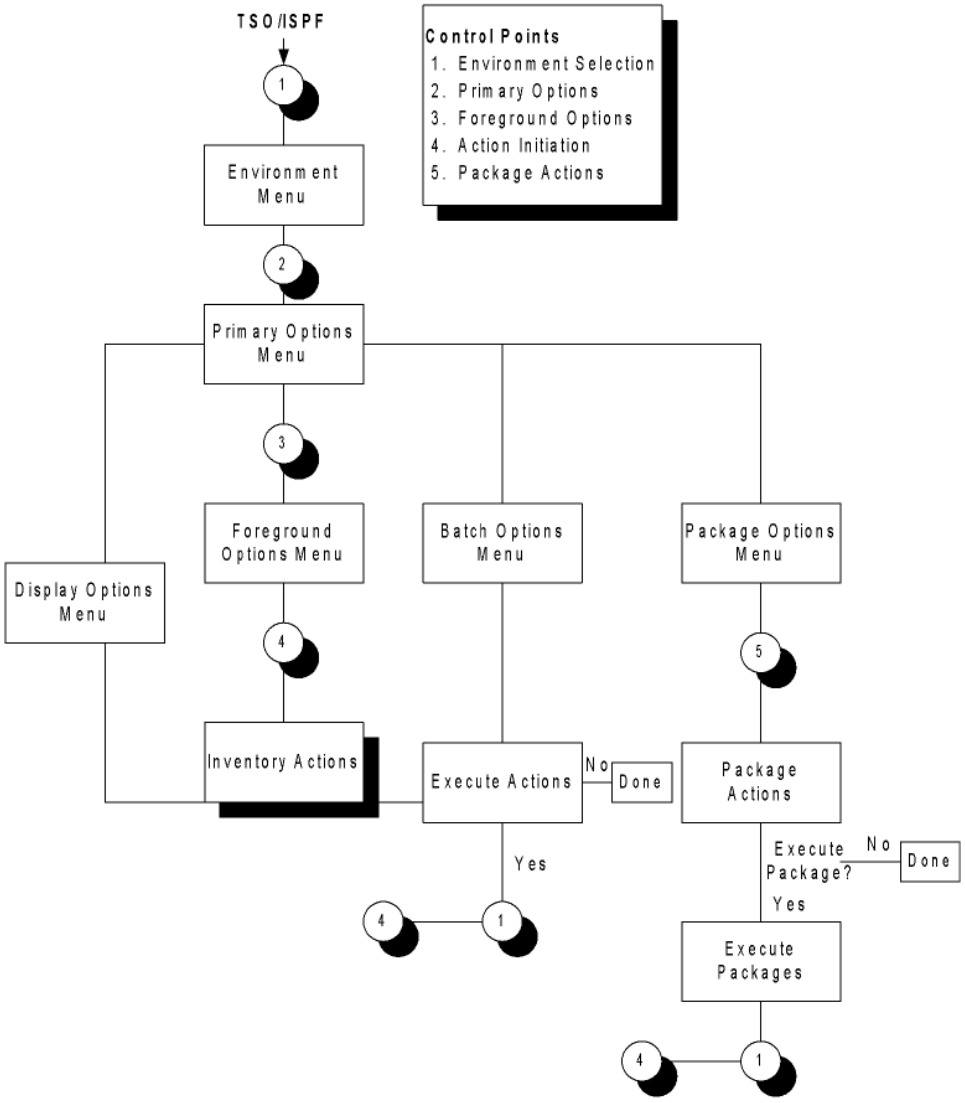
Important! To prevent user access to specific systems/subsystems or environments, you must define the Access Security Table and the User Security Table. To prevent unauthorized access to elements and actions, you must explicitly define the Resource Security Table. Additional security checking is available through User Exit 1, described in the *Exits Guide*.

Native security tables are checked by the system at a series of security control points. The following sections describe the role of security control points in the CA Endevor SCM processing flow.

Security Control Points

Security control is handled automatically through a series of security control points. Each control point invokes a routine that checks access and user privileges defined in the native security tables.

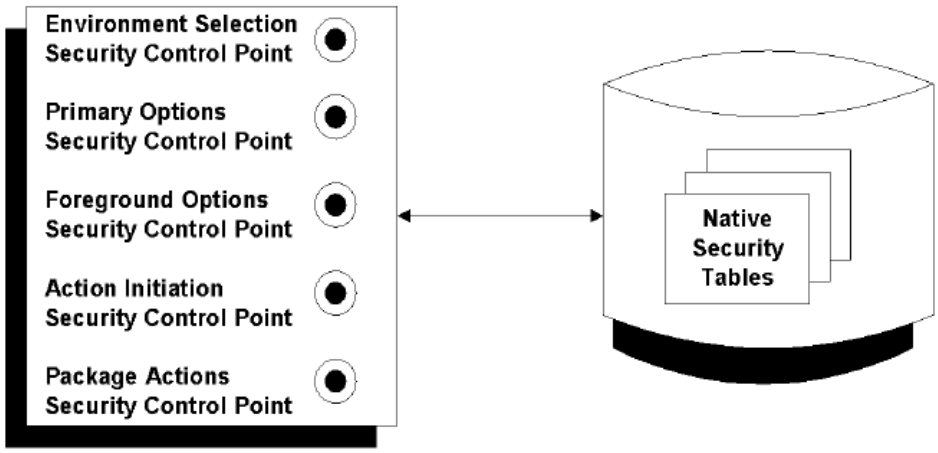
The following figure illustrates the strategic locations of security control points in the CA Endevor SCM processing flow. Each security control point controls access to functions and inventories occurring below it.



For example, the Environment Selection security control point controls access to the Environment menu and the Primary Options security control point controls access to the Primary Options menu.

Note the Environment Selection security control point appearing beneath the Batch Options menu. It regulates access to elements and functions available through the Batch Options and Package Options menus.

The following figure illustrates that, at each control point, CA Endevor SCM checks the native security tables to establish the access privileges defined for a particular user, environment, or system/subsystem.



The Environment Selection Security Control Point

The Environment Selection security control point (also referred to as the Environment Access security control point) checks the Access Security Table to determine environment access privileges. This security check occurs prior to building the Environment Selection menu used to gain access to an environment.

The Primary Options Security Control Point

The Primary Options security control point checks the User Security Table to determine a user's primary option privileges. These privileges include the Defaults, Display, Foreground, Batch, Environment, and Tutorial options. This security check occurs prior to building the Primary Options menu for an environment after access has been granted through the Environment Selection menu.

The Foreground Options Security Control Point

The Foreground Options security control point checks the User Security Table to determine a user's foreground options. These privileges include Display, Add/Update, Retrieve, Generate, Move, Delete, Print, and Signin options. This security check occurs prior to building the Foreground Options menu for an environment.

The Action Initiation Security Control Point

The Action Initiation security control point occurs:

- Prior to a cast operation during package processing, if the Defaults Table PKGCSEC flag is set to Y.
- Prior to an inspect operation during package processing, if the Defaults Table PKGISEC is set to Y.
- During package verification processing.
- Prior to performing a CA Endevor SCM action.

The Action Initiation security control point checks the User Security Table to determine a user's action privileges as defined by the user's access level. These privileges include CA Endevor SCM actions, such as ADD, RETRIEVE, GENERATE, MOVE, DISPLAY, ALTER, ARCHIVE, RESTORE, COPY, and LIST, as well as the SIGNOUT OVERRIDE, and ALL options.

The Package Actions Security Control Point

The Package Actions security control point occurs prior to performing the requested package action.

Note: The actions used to create and use packages are not controlled by native security, but can be controlled by ESI.

Defining User Exit Modules

You can define a user exit module at exit point 1 to do the following:

- Supplement the menu-building checks the system makes at each security control point.
- Supplement the action-request authorization that occurs at each security control point.

Exit 1 can only further restrict security; it cannot override restrictions imposed by your site security package.

Note: For more information about exits, see the *Exits Guide*.

How CA Endevor SCM Reads the Security Tables

When verifying access to site components, CA Endevor SCM uses the first applicable entry it encounters in each security table.

User Access

To verify a user's access to an environment, CA Endevor SCM checks the Access Security Table for a userid entry that applies for that user. A match to the requesting user ID (user ID 7, for example) could take the form of a full mask ($$$$$$$$), a partial name (USER$), or an exact match on the name (user ID 7) for the User Security and Resource Security tables. For the Access Security Table, you can use a single mask character ($) to specify a full mask for system and subsystem security. CA Endevor SCM uses the most restrictive specification that matches the user ID to validate the access.

System or Subsystem Access

To verify a user's access to a particular system and subsystem, CA Endevor SCM checks the User Security Table for the first entry that applies for that user, again matching to a full mask ($), a partial name (USER$), or an exact match on the name (user ID 7). It uses the specifications defined by this entry to validate the access.

Resource Restrictions

To check for restrictions that apply when performing actions, CA Endevor SCM looks in the Resource Security Table for an entry that applies to the element specified for the current action (CAELE1, for example). The match by element (resource) name is based on a full mask ($), a partial name (C$), or an exact match to the name (CAELE1).

If CA Endevor SCM does not find a table entry for the element, processing continues. If it does find a table entry for the element, CA Endevor SCM checks to see if that entry includes the system and subsystem specified in the action, matching the table entry's system/subsystem name based on a full mask, a partial name, or an exact match. If the entry includes the system/subsystem for the action request, CA Endevor SCM allows the action if the access level required for the action is specified in the table entry.

If the appropriate access level is not specified *explicitly* in the table entry, CA Endevor SCM does not allow the action.

How to Implement Native Security

To implement native security

1. Define the Access, User, and Resource Security Tables to control access to the environments, systems/subsystems, and elements you want to protect from unauthorized access.
2. Assemble and link-edit each security table you have defined.

3. Update the Defaults Table to reference the security tables you have defined.
4. Once you have sufficiently tested your functional security, copy the security table you defined as well as the new Defaults table, into an authorized library and perform an IPL or LLA refresh.

Note: If you are testing your security system, you do not need to issue an LLA refresh each time you edit a native security table if the tables are not in an authorized linklist library. Once you are ready for production mode, you need to copy your security tables into an authorized linklist library for security, and issue an LLA refresh.

The following sections explain each step in greater detail.

Define Your Native Security Tables

To enable CA Endevor SCM native security, you must define the native security tables to control access to the inventories and functions you want to protect.

Important! If you do not define the native security tables, these functions are not protected.

To define a native security table

1. Access the native security table you want to define.
2. Enter the appropriate CONSDEF macros.

   The native security table is defined.
3. Assemble and link-edit the table.
4. Update the Defaults Table.
5. Perform an LLA refresh if the table is in an authorized linklist library.

Enter CONSDEF Macros

You use a series of CONSDEF macros to define each security table. Once the CONSDEF macros have been assembled and link-edited successfully for each table, the table definition is complete. The table goes into effect once the Defaults Table is updated. Instructions for updating the Defaults Table appear in Modifying the Defaults Table.

Before coding the CONSDEF macros, you should understand:

- How to use coding conventions
- How to specify a mask
- How CA Endevor SCM reads security definitions

Note: For more information about defining each security table, see How to Define the Access Security Table (see page 39).

Note: If a user is in CA Endevor SCM while a security table is being updated or edited, the administrator must exit and reenter CA Endevor SCM in order for the new table to take effect.

Using Coding Conventions

When coding CONSDEF macros, follow the standard IBM rules:

- Place all macro specifications between columns 2 and 71 of the definition deck. To continue across cards, enter an **X** in column 72 and start the continuation card in column 16.
- Include at least one space between the macro name, CONSDEF, and the first keyword parameter, TYPE. Do not include any more spaces. An exception applies for literal operands, where spaces can be included between the surrounding (single) quotes.
- Specify each macro keyword fully. You cannot use a shortened version of any keyword.

Order of Security Definitions

CA Endevor SCM uses the most restrictive parameters to determine security checks for each user (Access Security Table and User Security Table) and each resource (Resource Security Table).

In the following example, users having IDs that start with the letters CADEV, except for the two exceptions, have access to all systems and subsystems, for all types of processing.

```
CONSDEF TYPE=USER,USERID=CADEV1,SYSDEF=((FINANCE,$,N))
CONSDEF TYPE=USER,USERID=CADEV8,SYSDEF=(($,$,N))
CONSDEF TYPE=USER,USERID=CADEV$,SYSDEF=(($,$,ADMPRSUZ,BV))
```

The two exceptions, CADEV1 and CADEV8, are restricted in the following ways:

- User CADEV1 can access all systems except FINANCE.
- User CADEV8 cannot access any system.

How to Define the Access Security Table

You use the Access Security Table to specify the environment(s) to which each user has access. You do this by specifying the unique Stage 1 name for the environment you want to access. There is one Access Security table for each installed site.

Note: Sample access security table source is provided in member ACCSTABL of installation library iprfx.iqual.CSIQSRC.

You define access to environment(s) either directly for each user ID, or for groups of users. Where you specify access for a group, each group must be associated with the specific user IDs included in the group (either before or after the group-level definition).

You might use the specification below, for example, to allow user DMS access to the stage names you specify; this allows user DMS access to the Stage 1 name in the environment in which it is located:

```
TYPE=USER,USERID=DMS,SYSDEF=((stage1-name,$,R),(stage1-name,$,R))
```

Or, you might permit access to the stage names you specify through group DVLP, then associate user DMS with that group:

```
TYPE=USER,GROUP=DVLP,SYSDEF=((stage1-name,$,R),(stage1-name,$,R))
TYPE=USER,USERID=DMS,GROUP=DVLP
```

To define the Access Security Table, enter CONSDEF macros in the following format:

```
 CONSDEF TYPE=START,TABLE=USER

Form 1:
 CONSDEF  TYPE=USER,USERID=user-id,SYSDEF=((stage1-name,$,R)...)


Form 2:
 CONSDEF  TYPE=USER,GROUP=group-name,SYSDEF=((stage1-name,$,R)...)


Form 3:
 CONSDEF  TYPE=USER,USERID=user-id,GROUP=group-name


 CONSDEF  TYPE=END,TABLE=USER
```

You can use any combination of these statements, but must start with the TYPE=START macro and end with the TYPE=END macro.

Note: It is essential that you assign each stage in each environment a unique name. Since you are specifying access to environments by way of stage name, you must be sure that each stage name is unique, across environments.

type=start,table=user

Defines the beginning of the Access Security Table definition.

type=end,table=user

Terminates the table definition.

type=user

Either defines the environment(s) accessible to a particular user or group (forms 1 and 2) or associates a user with a group-level access definition (form 3). The TYPE=USER macros can be coded in any order. Be sure, however, that each group associated with a user (form 3) has a corresponding group access definition (form 2).

A user must be given explicit permission through the TYPE=USER statements to process in a CA Endevor SCM environment.

type=user Parameters

Each TYPE=USER parameter is described next.

userid=userid

Defines the users for which access information is being specified (form 1) or, if used with a GROUP specification (form 3), the ID of the users associated with the group-name. The userid can end with a mask character to include all users having IDs that start with the characters specified or it can be specified as eight mask characters, $$$$$$$$ to include all users. The following parameter specifies all users having IDs that start with AN:

```
USERID=AN$
```

If your site has userids containing a $ you can change the $ mask to another character. Edit the iprfx.iqual.CSIQOPTN library member MASKMAC and change the &CHAR=$ to another character. In the following example the mask character is set to @:

```
MASKMAC &STRIP=NO,&CHAR=@,&MAXLTH=8
```

group=group-name

Defines the group (1-8 characters) for which access information is being specified (form 2) or, if used with a user ID specification (form 3), the group associated with the userid specified. Group names are specific to this macro; they are not referenced elsewhere within CA Endevor SCM. A group name might be, for example, PAYROLL, QA, DBA, SYSTEMS, and so forth. Once a group is defined, all users associated with the group acquire access to the environment(s) defined for the group.

sysdef=((stage1-name,$,r))

Specifies an environment to which the user (form 1) or group (form 2) has access. The environment is defined in terms of the stage1-name defined for the environment. We recommend using different stage names for each environment. The second and third positional parameters within the SYSDEF specification must be the characters $ and R, respectively.

The mask character is supported for stage1-name. Generally, however, it is easier to repeat the operands within the parentheses when defining access to multiple environments, as shown below:

```
SYSDEF=((stage-1-name,$,R),(stage-1-name,$,R))
```

Assemble and Link-Edit the Access Security Table

You can use an SMP/E USERMOD to assemble and link-edit ACCSTABL after it has been customized. Alternatively, you can edit the sample JCL BC1JTABL and use it to assemble and link source module ACCSTABL outside of SMP/E. BC1JTABL is supplied in the installation library iprfx.iqual.CSIQJCL.

After you have defined the Access Security Table, verify that the JCL is correct and run the job to assemble and link-edit the new table. The name of the table is specified as the SYSLMOD member name in the JCL.

The output load module for the table is placed in the LOADLIB established during installation.

Note: For testing purposes, you can copy the Access Security Table to a standard load library. Once you are ready to go into production mode, be sure to copy the load module to an authorized LINKLIST library. After a successful link-edit, update the Defaults Table to point to the new Access Security Table. Set the ACCSTBL Defaults Table parameter in the TYPE=MAIN macro to identify the SYSLMOD DD member name assigned in the JCL.

How to Define the User Security Table

The User Security Table specifies the system(s) and subsystem(s) to which each user has access, within a particular environment. For each system/subsystem to which a user has access, the table also specifies the type(s) of processing allowed (retrieve-only, add, update, and so forth). There is (at most) one User Security Table for each environment.

Note: Sample user security table source is provided in member USERTABL of installation library *iprfx.iqual.CSIQSRC*.

You can define access to the system(s)/subsystem(s) either directly for each userid or for groups of users. Where you specify access at the group level, each group must be associated with the specific user IDs included in the group.

You might use the following specification, for example, to allow user DMS access to the Finance and GL systems (all subsystems). The user can access the Finance system to move (M), generate (P), or display (B) elements, and the GL system to display only.

```
TYPE=USER,USERID=DMS,SYSDEF=((FINANCE,$,MP,B),(GL,$,,B))
```

Alternatively, you might permit this access through group ACCT, then associate user DMS with that group:

```
TYPE=USER,GROUP=ACCT,SYSDEF=((FINANCE,$,MP,B),(GL,$,,B))
TYPE=USER,USERID=DMS,GROUP=ACCT
```

Enter the CONSDEF macros using the following format to define the User Security Table:

```
 CONSDEF TYPE=START,TABLE=USER

Form 1:
 CONSDEF TYPE=USER,USERID=user-id,[UNTIL=yyddd,]                      X
                SYSDEF=((sys-name,subsys-name,access1,access2)...

Form 2:
 CONSDEF TYPE=USER,GROUP=group-name,[UNTIL=yyddd,]                    X
                SYSDEF=((sys-name,subsys-name,access1,access2)...

Form 3:
 CONSDEF TYPE=USER,USERID=user-id,GROUP=group-name

 CONSDEF TYPE=END,TABLE=USER
```

You can use any combination of these statements, but they must start with the TYPE=START macro and end with the TYPE=END macro.

type=start,table=user

Defines the beginning of the User Security Table definition.

type=end,table=user

Terminates the table definition.

type=user

Either defines the system(s)/subsystem(s) accessible to a particular user or group (forms 1 and 2) or associates a user with a group-level access definition (form 3). The type=user macros can be coded in any order. Make sure, however, that each group associated with a user (form 3) has a corresponding group access definition (form 2).

In order for a user to have access to a particular system/subsystem configuration, that user must be given explicit permission to process within that configuration, for the type of access desired.

type=user Parameters

Each TYPE=USER parameter is described next:

userid=user-id

Defines the user(s) for which access information is being specified (form 1), or, if used with a group specification (form 3), the ID of the user(s) associated with the group-name. The user-id can end with a mask character to include all users having IDs that start with the characters specified, or it can be specified as eight mask characters, $$$$$$$$, to include all users. The following parameter specifies all users with IDs that start with AN:

```
USERID=AN$
```

If your site has userids containing a $ you can change the $ mask to another character. Edit the iprfx.iqual.CSIQOPTN library member MASKMAC and change the &CHAR=$ to another character. In the following example the mask character is set to @:

```
MASKMAC &STRIP=NO,&CHAR=@,&MAXLTH=8
```

group=group-name

Defines the group (up to eight characters) for which access information is being specified (form 2) or, if used with a user ID specification (form 3), the group associated with the user-id specified. Group names are specific to this macro; they are not referenced elsewhere within CA Endevor SCM. A group name might be, for example, Payroll, QA, DBA, Systems, and so forth. Once a group is defined, all users associated with the group acquire access to the configuration(s) defined for the group.

[until=yyddd]

Defines the (Julian) date through which the defined access is valid. This parameter is optional. If omitted, there is no expiration on the definition.

sysdef=((sys-name,subsys-name,access1,access2)...)

Specifies a system/subsystem/access-level configuration to which the user (form 1) or group (form 2) has access. The configuration is defined using four positional parameters:

sys-name

The CA Endevor SCM system to which the user has access (restricted according to the *subsys-name* and access codes specified).

subsys-name

The name of the subsystem to which the user has access within the system named as option 1 (and restricted according to the access codes).

access1

A list of single-character codes that identify the types of access for which the user is authorized within the system and subsystem named. Do not include a separator character between multiple codes within each access specification. The *access1* parameter specifies all types of access except Display and Archive, which must be specified using *access2*. If you want to specify Display or Archive access only, include a positional comma for *access1*. If you want neither Display nor Archive access, you need not include a final positional comma.

access2

A list of single-character codes that identify the types of access for which the user is authorized within the system and subsystem named. Specify Display and Archive using *access2*.

The following are access codes for the *access1* and *access2* variables.

Access Code	*access1*	*access2*	Access Level
A	X		Add
B		X	Display
D	x		Delete
M	x		Move
P	x		Generate
R	x		Retrieve
S	x		Signout override
U	x		Update
V		x	Archive
Z	x		Environment administration
N	x		No access

The *sys-name* and *subsys-name* variables can end with a mask character (for example, DEV$), to include all systems/subsystems starting with the same characters; or you can use a single mask character to include all systems/subsystems. For example, the following statement specifies add-access for all subsystems within systems beginning with the characters GL:

```
SYSDEF=((GL$,$,A))
```

To allow access to multiple configurations, repeat the operands within the parentheses:

```
SYSDEF=((sys,subsys,access,access),(sys,subsys,access,access)...)
```

The following table describes the access levels necessary for various types of CA Endevor SCM processing. Note that access levels for moves and transfers appear in a separate table that follows this table.

Action	Access Level	Access Code
Add	Add	A
Update	Update	U
Retrieve	Retrieve	R
Generate (Stage 1 or Entry Stage)	Generate	P
Generate (Stage 2 non-entry)	Move	M
Display	Retrieve or Display	R or B
Delete (Stage 1 or Entry)	Delete	D
Delete (Stage 2 non-entry)	Move	M
Signin	Retrieve	R
Print	Retrieve or Display	R or B
Alter (stage 1 or Entry stage)	Update	U
Alter (Stage 2 non-entry)	Move	M
Archive	Archive	V
Restore to Stage 1 or Entry Stage	Add	A
Restore to Stage 2 non-entry	Move	M
Copy	None	
List	Display	B
Signout override	Signout Override	S
All (element type Process)	Environment management	Z
Checks while building the primary Options Menu:		
Include Defaults option	None	
Include Display option	Retrieve or Display	R or B
Include Foreground option	Retrieve	R
Include Batch option	Retrieve	R
Include Environment option	Environment management	Z
Include Tutorial option	None	

Action	Access Level	Access Code
Checks while building the Foreground Options menu:		
Include Option	None	—
Include Add/Update option	Add or Update	A or U
Include Retrieve option	Retrieve	R
Include Generate option	Generate	P
Include Delete option	Delete	D
Include Print option	Retrieve or Display	R or B
Include Signin option	Retrieve	R

Moves and transfers often involve combined actions and source and target considerations that require special access codes. The following table provides the access codes required for performing moves and transfers according to stage, source, and target parameters.

Action	Access Level	Access Code
Moves From:		
Stage 1, Stage 2, and entry stage	Move	M
Moves To:		
Stage 1 or Entry Stage	Add	A
Stage 2 non-entry	Move	M
Transfers From:		
Stage 1 or Entry Stage, no Delete	Retrieve	R
Stage 1 or Entry Stage, with Delete	Delete	D
Stage 2 non-entry no Delete	Retrieve	R
Stage 2 non-entry with Delete	Move	M
Transfers To:*		
Stage 1 or Entry Stage	Add	A
Stage 2 non-entry	Move	M

***Transfers to a target destination do not permit a delete.

How to Assemble and Link-Edit the User Security Table

Use an SMP/E USERMOD to assemble and link-edit your user security table. Alternatively, you can edit the sample JCL BC1JTABL and use it to assemble and link-edit your table outside of SMP/E. BC1JTABL is located in the installation library *iprfx.iqual.CSIQJCL*. BC1JTABL includes a step with a default name of USERTABL for the user security table. Verify that this name and the rest of the JCL is correct before submitting this job.

This assembles and link-edits the new table. The name of the new table is specified as SYSLMOD member name in the JCL.

The output load module for the table is placed in the LOADLIB established during installation.

Note: For testing purposes, you can copy the User Security Table to a standard load library. Once you are ready to go into production mode, be sure to copy the load module to an authorized LINKLIST library.

After a successful link-edit

1. Update the Defaults Table (C1DEFLTS).

   This will point to the new User Security Table.

2. Set the USERTBL parameter in the TYPE=ENVRNMNT macro to

   This will identify the SYSLMOD DD member name assigned in the JCL.

Note: See How to Modify the Defaults Table (see page 52) for a brief description of this procedure.

How to Define the Resource Security Table

The Resource Security Table defines element names that are restricted to a particular system(s)/subsystem(s), within a particular environment. For each restricted element name, the table also specifies the type(s) of processing for which the restriction applies (retrieve-only, add, update, and so forth). There is one Resource Security Table for each environment.

Note: Sample resource security table source is provided in member RSCTTABL of installation library iprfx.iqual.CSIQSRC.

You can define resource security directly for each resource (element name), or you can define it for groups of resources. Where you specify security at the group level, each group must be associated with the specific element names included in the group.

For example, you would use the specification below to indicate that any element names starting with the characters ACCT can only be added to the ACCOUNTS subsystem of the FINANCE system:

```
TYPE=RESOURCE,RNAME=ACCT$,SYSDEF=((FINANCE,ACCOUNTS,A))
```

This does not limit the ACCOUNTS subsystem (FINANCE system) to element names starting with the characters ACCT, but simply prevents these element names from being added to other systems/subsystems.

If your site has element names containing a $ you can change the $ mask to another character. Edit the iprfx.iqual.CSIQOPTN library member MASKMAC and change the &CHAR=$ to another character. In the following example the mask character is set to @:

```
MASKMAC &STRIP=NO,&CHAR=@,&MAXLTH=8
```

To illustrate a group-level specification, you would use the following definition to indicate that element names beginning with the characters AP or AR can only be added to the FINANCE or GL systems, any subsystem:

```
TYPE=RESOURCE,GROUP=XYZ,SYSDEF=((FINANCE,$,A),(GL,$,A))
TYPE=RESOURCE,RNAME=AP$,GROUP=XYZ
TYPE=RESOURCE,RNAME=AR$,GROUP=XYZ
```

To define the Resource Security Table, enter CONSDEF macros in the following format:

```
CONSDEF TYPE=START,TABLE=RESOURCE

Form 1:
CONSDEF  TYPE=RESOURCE, RNAME=element-name,[UNTIL=yyddd,]            X
             SYSDEF=((sys-name,subsys-name,access1,access2)...)

Form 2:
CONSDEF TYPE=RESOURCE,GROUP=group-name,[UNTIL=yyddd,]                X
             SYSDEF=((sys-name,subsys-name,access1,access2)...)
Form 3:
CONSDEF TYPE=RESOURCE,RNAME=element-name,GROUP=group-name

CONSDEF     TYPE=END,TABLE=RESOURCE
```

You can use any combination of these statements, but they must start with the TYPE=START macro and end with the TYPE=END macro.

type=start,table=resource

Defines this as the beginning of the Resource Security Table definition.

type=end,table=resource

Terminates the table definition.

type=resource

Defines an element name(s) that is restricted to a particular system and subsystem (form 1) or a group of element names that are restricted to a particular system and subsystem (form 2) or associates an element name with a group-level definition (form 3). The TYPE=RESOURCE macros can be coded in any order. Make sure, however, that each group associated with an element name (form 3) has a corresponding group definition (form 2).

In order for a restriction to apply for an element name, that name must be associated with the type(s) of processing (access levels) for which the restriction applies (retrieve-only, add, update, and so forth).

type=user Parameters

Each TYPE=RESOURCE parameter is described next:

rname=element-name

Defines the element name(s) for which a restriction is being specified (form 1) or, if used with a group specification (form 3), the element name(s) associated with the group-name. The element-name can end with a single mask character to include all element names that start with the characters specified, or it can be specified with eight mask characters, ($$$$$$$$), to include all element names. The following parameter specifies all element names that start with the characters FIN:

```
RNAME=FIN$
```

group=group-name

Defines the group (up to eight characters) for which a restriction is being defined (form 2), or, if used with an RNAME specification (form 3), the group associated with the element-name specified. Group names are specific to this macro; they are not referenced elsewhere within CA Endevor SCM. A group name might be, for example, Payroll, QA, DBA, Systems, and so forth. Once a group is defined, all element names associated with the group acquire the restrictions defined for the group.

[until=yyddd]

Defines the (Julian) date through which the defined restriction is valid. This parameter is optional. If omitted, there is no expiration on the definition.

sysdef=((sys-name,subsys-name,access1,access2)...)

Specifies the system/subsystem configuration to which the element name (form 1) or group (form 2) is restricted. The configuration is defined using four positional parameters:

sys-name

The name of the CA Endevor SCM system to which the element name is restricted. You can use a single mask character ($) to specify all system names.

subsys-name

The name of the subsystem to which the element name is restricted, within the system named. You can use a single mask character ($) to specify all subsystem names.

access1

A list of single-character codes that identify the types of access under which the element name is restricted, within the system and subsystem named. Within each access specification, do not include a separator character between multiple codes. The parameter *access1* specifies all types of access except Display and Archive, which must be specified using *access2*.

access2

A list of single-character codes that identify the types of access under which the element name is restricted, within the system and subsystem named.

If you want to specify Display or Archive access only, include a positional comma for *access1*. If you do not want Display or Archive access, you do not need to include a final positional comma.

Note: For more information about the correct access codes, see How to Define the User Security Table (see page 42).

The *sys-name* and *subsys-name* variables can end with a mask character (for example DEV$) to include all systems/subsystems that start with the characters specified, or they can be specified as a single mask character to include all systems/subsystems.

To restrict element names to any of several configurations, repeat the operands within the parentheses:

```
SYSDEF=((syse,subsys,access,access),(sys,subsys,access,access)...)
```

How to Assemble and Link-Edit the Resource Security Table

Use a SMP/E USERMOD to assemble and link-edit your resource security table. Alternatively, you can edit the sample JCL BC1JTABL to assemble and link-edit your table outside of SMP/E. BC1JTABL is located in the installation library iprfx.iqual.CSIQJCL. BC1JTABL has a step, with a default name of RSCTTABL, for the resource security table. Make sure that this name and the rest of the JCL is correct before submitting this job.

This assembles and link-edits the new table. The name of the new table is specified as SYSLMOD member name in the JCL.

The output load module for the table is placed in the LOADLIB established during installation.

Note: For testing purposes, you can copy the User Security Table to a standard load library. Once you are ready to go into production mode, be sure to copy the load module to an authorized LINKLIST library. See the *Implementation Guide* for a complete description of this procedure.

After a successful link-edit

1. Update the Defaults Table (C1DEFLTS).

   This will point to the new User Security Table.

2. Set the RSCETBL parameter in the TYPE=ENVRNMNT macro to identify the SYSLMOD DD member name assigned in the JCL.

How to Modify the Defaults Table

To enable native security

- Modify the Defaults Table to reference the security tables you have defined.
- Copy the Defaults Table into an authorized load library.
- This step should be undertaken only *after* you have completed testing your security definitions.
- Perform an LLA refresh.

To modify the Defaults Table

1. Specify the following parameters in the Defaults Table.

TYPE=MAIN Parameters

ACCSTBL

A name (up to eight characters) of the Access Security Table in use at your site.

TYPE=ENVRNMN: Parameters:

USERTBL

A name (up to eight characters) of the User Security Table for this environment.

RSCETBL

A name (up to eight characters) of the Resource Security Table for this environment.

Use an SMP/E USERMOD to assemble and link-edit C1DEFLTS after it has been customized. Alternatively, edit the sample JCL BC1JTABL and use it to assemble and link source module C1DEFLTS outside of SMP/E. BC1JTABL is located in the installation library iprfx.iqual.CSIQJCL. This stores the defaults table in uprfx.uqual.CSIQAUTU as member C1DEFLTS.

Note: For more information about updating and modifying the Defaults Table, and about load libraries, see the *Administration Guide*.

Chapter 4: Enabling External Security Interface

This section contains the following topics:

How ESI Security Works

The CA Endevor External Security Interface (ESI) is an option that unifies security for CA Endevor SCM and your site security package (RACF, CA ACF2 for z/OS, or CA Top Secret for z/OS). ESI allows you to do the following:

- Extend your site security package to control and authorize access to components maintained by CA Endevor SCM.
- Secure user actions ranging from environment access to specific action checks.
- Customize ESI security through a table-driven architecture.

ESI converts combinations of CA Endevor SCM entities such as an environment name or an element name into pseudo data set names and then queries your site security package for a ruling whether or not the user is allowed to access the data set. Examples of CA Endevor SCM entities that you can map into your site security package rules are:

- Environments
- Elements
- Stages
- CCIDs
- Actions
- Systems/subsystems

Note: Each node in a data set name only allows a maximum of eight characters per value; therefore, values greater than eight characters are truncated to eight characters (e.g., EMERGENC).

ESI uses CA Endevor SCM provided security checkpoints (Exit 01) to determine whether to allow or deny a user access to an inventory, environment or a specific function/action. ESI processing is invoked at all security control points.

ESI uses a compiled table called the Name Equates Table to specify rules that map entity names to pseudo data set names when a security control point is encountered. ESI then constructs the pseudo data set name and executes the operating system's RACROUTE macro.

The RACROUTE macro provides an application with access to the System Authorization Facility (SAF) which in turn communicates with your site security package (RACF, CA ACF2 for z/OS, or CA Top Secret for z/OS). SAF allows CA Endevor SCM to request authorization information for any site security package.

The Name Equates Table consists of a set of ESI Defaults entries (ESIDFLTS), Function Equates entries (FUNCEQU) and Name Equates entries (NAMEQU) described in the following sections.

ESI Defaults Entries (ESIDFLTS)

The ESIDEFLTS entry allows you to establish the default behavior for all security call formats and tracing. It also includes an option to improve performance.

Note: For more information about the defaults entry, see How to Define ESI Diagnostics (see page 62).

Function Equates Entries (FUNCEQU)

Each FUNCEQU entry allows you to map CA Endevor SCM actions to authorization values or keywords that are understood by your site security package.

Note: For more information, see Map Authorization Values (see page 66).

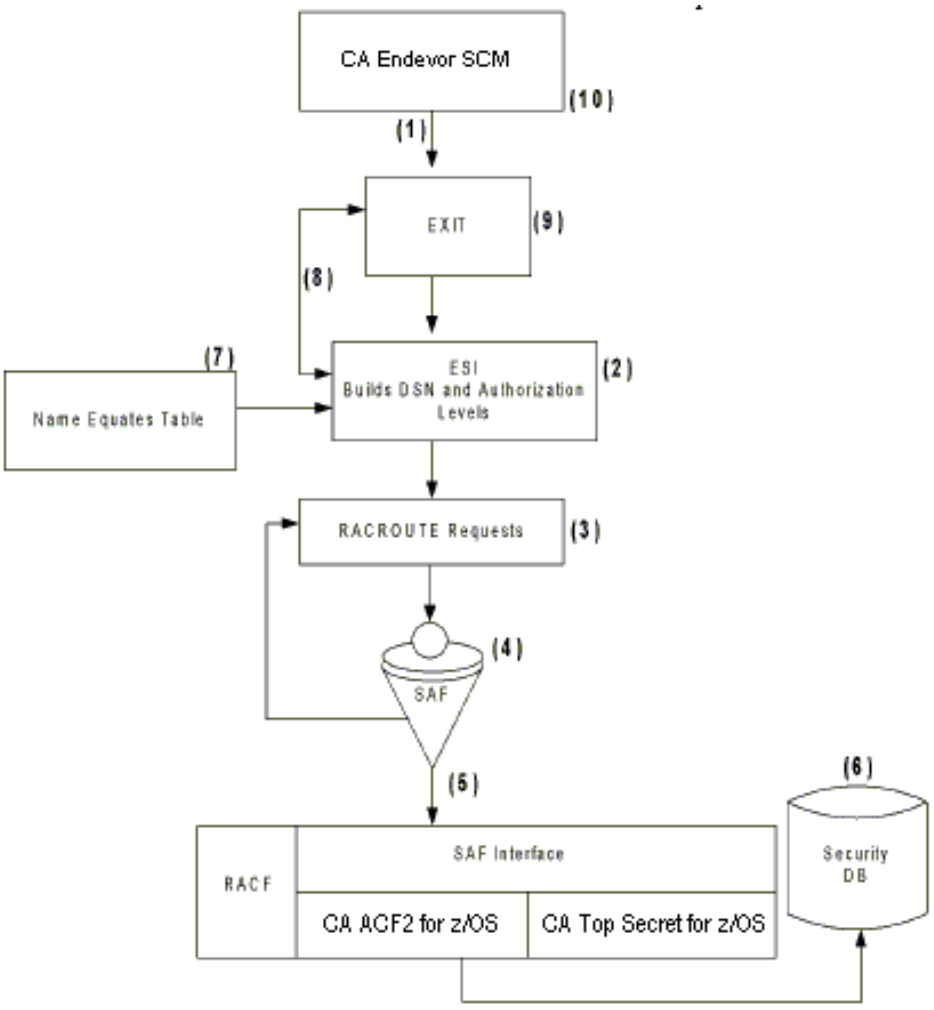
Name Equates Entries (NAMEQU)

The NAMEQU entries become the pseudo data set names that are sent to the RACROUTE macro. Each entry corresponds to a specific security control point that you can secure. There are six different security call formats for pseudo data sets. The following shows the correlation of the pseudo data set formats to security control points:

Security Control Point	Format	Where the Security Check Occurs
Environment	ENVIRONMENT_ACCESS (formerly FORMAT1)	■ Prior to building the Environment Selection menu ■ When you change the current environment through a panel ■ During batch processing to validate your environment access
Primary Options	PRIMARY_OPTIONS (formerly FORMAT2)	■ Prior to building the Environments Primary Options menu ■ During LOAD/UNLOAD/RELOAD processing ■ During batch package processing
Foreground Options	FOREGROUND_OPTION (formerly FORMAT3)	■ Prior to building the Foreground Options menu for the environment
Action Initiation	ACTION_INITIATION-standard and extension (formerly FORMAT4 and FORMAT5)	■ Prior to performing the requested action ■ Prior to cast, during package processing if PKGCSEC=Y ■ Prior to inspect, during package processing if PKGISEC=Y ■ During package verification processing
Package Actions	PACKAGE_ACTIONS	■ Prior to performing the requested package action
Concurrent Action Processing	CONCURRENT_ACT_PROC	■ Prior to initiating concurrent batch actions

The Security Processing Model

The following figure shows how CA Endevor SCM ESI works with a site security package such as RACF, CA ACF2 for z/OS, or CA Top Secret for z/OS.



The following list describes the security processing model illustrated in the previous figure.

1. CA Endevor SCM calls Exit 01 processing, which calls ESI.
2. Exit01 calls ESI.
3. ESI constructs the data set name (DSN) and authorization level from the Name Equates Table.
4. ESI issues a RACROUTE request with the DSN and authorization levels. RACROUTE requests are then routed to the System Authorization Facility (SAF).

5. SAF routes the RACROUTE request directly to the site security package.
6. The security package interprets the request by looking up the request in the security database.
7. SAF returns to ESI and the request either passes or fails.
8. ESI returns to Exit 01 processing.
9. CA Endevor SCM Exit 01 processing runs other user exits.
10. Exit 01 processing may return to CA Endevor SCM.

User Exit Modules

You can define a user exit module at exit point 1 to do the following:

- Supplement the menu-building checks the system makes at each security control point.
- Supplement the action-request authorization that occurs at each security control point.

Note: Exit 01 can only further restrict security, it cannot override restrictions imposed by your site security package (RACF, CA ACF2 for z/OS, or CA Top Secret for z/OS).

Name Equates Table

The Name Equates Table allows you to specify security rules that map CA Endevor SCM entity names to pseudo data set names. The following is a sample Name Equates Table:

```
         TITLE 'BC1TNEQU - EXTERNAL SECURITY INTERFACE TABLE.'
*************************************************************************
*        DEFINE ESI DEFAULTS                                            *
*************************************************************************
BC1TNEQU ESIDFLTS WARN=NO,                                               +
               TITLE='BC1TNEQU SECURITY INTERFACE TABLE',                +
               HEADER=YES,                                               +
               LATSIZE=2,                                                +
               DESC=6,                                                   +
               ROUTCDE=11
*************************************************************************
*        MAP E/MVS AUTHORITIES TO SAF AUTHORITIES FOR                   *
*        ACTION_INITIATION AND PACKAGE_ACTIONS FORMAT CALLS.            *
*        NOTE: ENVIRONMENT_ACCESS, PRIMARY_OPTIONS AND                  *
*              FOREGROUND_OPTIONS FORMAT CALLS ALWAYS USE READ          *
*              AUTHORITY AND CANNOT BE MODIFIED.                        *
*************************************************************************
         FUNCEQU SAFAUTH=READ,                                           +
               C1ACTNS=(ADD,ARCHIVE,DELETE,                              +
               DISPLAY,ENVRNMGR,GENERATE,MOVE,                           +
               PBACKOUT,PCAST,PCOMMIT,PCREATE,PDISPLAY,PDYNAMIC,         +
               PEXECUTE,PLIST,PMODIFY,PREVIEW,PSHIP,                     +
               PUTILITY,RETRIEVE,SIGNIN,SIGNOVR,UPDATE,VALIDATE)
*************************************************************************
*        SAMPLE SYNTAX OF OTHER SUPPORTED FUNCEQU AUTHORITIES LEVELS    *
*************************************************************************
*        FUNCEQU SAFAUTH=NONE,                                          +
*        FUNCEQU SAFAUTH=UPDATE,                                        +
*        FUNCEQU SAFAUTH=CONTROL,                                       +
         FUNCEQU SAFAUTH=ALTER,                                          +
               C1ACTNS=(ALTER)
*************************************************************************
*        END OF FUNCEQU SECTION                                         *
*************************************************************************
         FUNCEQU TYPE=END
         SPACE 2
*************************************************************************
*        SPECIFY SAF DATASET NAME FORMATS                               *
*************************************************************************
         NAMEQU ENVIRONMENT_ACCESS,                                      +
               L1=('C1'),                                                +
               L2=('ENVIRON'),                                           +
               L3=(ENVIRONMENT)
         NAMEQU PRIMARY_OPTIONS,                                         +
               L1=('C1'),                                                +
```

```
                L2=(ENVIRONMENT),                                      +
                L3=('PMENU'),                                          +
                L4=(MENUITEM)
          NAMEQU FOREGROUND_OPTIONS,                                   +
                L1=('C1'),                                             +
                L2=(ENVIRONMENT),                                      +
                L3=('FORACTN'),                                        +
                L4=(MENUITEM)
          NAMEQU ACTION_INITIATION,                                    +
                L1=('C1'),                                             +
                L2=(ENVIRONMENT),                                      +
                L3=(SYSTEM),                                           +
                L4=(SUBSYSTEM)
          NAMEQU ACTION_INITIATION,                                    +
                L1=('C1'),                                             +
                L2=(MENUAUTH)
          NAMEQU PACKAGE_ACTIONS,                                      +
                L1=('C1'),                                             +
                L2=('PACKAGE'),                                        +
                L3=(MENUITEM),                                         +
                L4=(PKGSUBFC),                                         +
                L5=(PKGID)
          NAMEQU CONCURRENT_ACT_PROC,                                  +
                CLASS='DATASET',                                       +
                WARN=NO,                                               +
                LOG=NONE,                                              +
                L1=('C1'),                                             +
                L2=('CAP')
*********************************************************************** *
*         SAMPLE SYNTAX OF OTHER SUPPORTED NAMEQU PACKAGE_ACTIONS       *
*         SYMBOLIC PARAMETERS                                           *
********************************************************************** ****
*               LN=(PKGAPPGR),                                          +
*               LN=(PKGBOE),                                            +
*               LN=(PKGSHR),                                            +
*               LN=(PKGSTAT),                                           +
*               LN=(PKGTYPE),                                           +
*               WARN=NO
*********************************************************************
*         END OF NAMEQU SECTION                                        *
*********************************************************************
          NAMEQU TYPE=END
          END
```

The following entries are coded in the Name Equates Table:

ESIDFLTS

This entry allows you to improve performance and establish the default behavior for all formats and traces.

FUNCEQU

This entry equates CA Endevor SCM access levels to authorization values for the RACROUTE attr=auth parameter.

NAMEQU

This entry creates the ENVIRONMENT_ACCESS through CONCURRENT_ACT_PROC pseudo data set names, which creates the entity=dsname value used by the RACROUTE request.

Code the ESIDFLTS, FUNCEQU and NAMEQU entries to:

- Improve performance and specify trace behaviors and diagnostic parameters (ESIDFLTS).
- Change SAF authorization levels (FUNCEQU).
- Change SAF pseudo data set name formats (NAMEQU).

The sections that follow describe how to code these entries to conform to your needs.

How to Define ESI Diagnostics

The ESI Defaults macro (ESIDFLTS) allows you to specify how you manage ESI diagnostics. You can use the ESIDFLTS macro to write a diagnostic trace, to improve performance and to enable warning mode.

Note: For more information about tracing and warning mode, see The ESI Trace Facility (see page 110) and The ESI Warning Mode (see page 109).

The security look aside table (LAT) feature allows you to reduce the number of calls to the SAF interface and thereby improve performance. The result of each resource access request to SAF is stored in the LAT. ESI checks the LAT first for authorization and if the resource access request is listed, ESI does not make a call to SAF.

Note: If your site uses the CA Endevor SCM for CA Roscoe Interface option, do not use the LAT feature in CA Endevor SCM for the CA Roscoe environment.

We recommend using the default settings in the sample ESIDFLTS macro included with your installation kit. You can define only one ESIDFLTS macro in the Name Equates table and you should list the ESIDFLTS macro first. The following excerpt from a sample Name Equates Table shows the ESIDFLTS macro with the LAT feature enabled.

```
ESIDFLTS TITLE='BC1TNEQU',
    HEADER=ALL,
    LATSIZE=5,
    WARN=NO
```

To turn on or off the LAT feature, code the ESIDFLTS entry according to the following syntax:

```
ESIDFLTS TITLE='string'
LATSIZE=n
```

In this syntax, TITLE specifies a character string defined in the Name Equates Table. LATSIZE specifies the number of 4K pages used to store access entries in the look aside table (LAT).

The format of the ESIDFLTS macro is listed next:

DESC=

Specifies the descriptor codes used with the Write to Operator (WTO) messages when the trace is written to the operator's console. WTO is the default if DESC= is not specified.

HEADER=(*ALL/NONE*)

Specifies whether to write the header information to the trace destination when opening the EN$TRESI DD. ALL (or YES) specifies that the header is written. NONE specifies that header information is not written.

LATSIZE=*n*

Specifies the number of 4K pages used to store access entries in the look aside table (LAT). There are approximately 35 entries per 4K page. We recommend a LATSIZE setting between 2 and 10. The default, LATSIZE=0, turns off the LAT. The maximum LATSIZE value is 524,287, which we do not recommended for use. When the LAT is full, new security calls are issued to SAF. The LAT size is allocated in each address space.

Note: The only method to determine if the LAT space gets full often and should be increased, is to check the ESI trace facility.

ROUTECDE=

Specifies the routing codes used with the Write to Operator (WTO) message when trace information is written to the operator's console. The default is WTO if ROUTCDE= is not specified.

TITLE=

Specifies a character string defined in the Name Equates Table. The title is displayed in the trace header. The default TITLE is 'No Title Specified'. The string 'BC1TNEQU' is added to the specified string.

WARN=(*YES/NO*)

Specifies warning mode for the entire table. Individual formats can override the WARN= setting.

How to Define SAF Authorization Levels

When ESI issues a RACROUTE request using ACTION_INITIATION at the Action Initiation security control point or PACKAGE_ACTIONS at the Package Actions security control point, it determines the SAF authorization level (attr=*auth*) by checking the FUNCEQU entry.

ENVIRONMENT_ACCESS, PRIMARY_OPTIONS, and CONCURRENT_ACT_PROC are always issued with a READ request and cannot be modified through the Name Equates table.

```
FUNCEQU SAFAUTH=READ,                                    +
   C1ACTNS=(ADD,ARCHIVE,DELETE,                          +
   DISPLAY,ENVRNMGR,GENERATE,MOVE,                       +
   PBACKOUT,PCAST,PCOMMIT,PCREATE,PDISPLAY,PDYNAMIC,     +
   PEXECUTE,PLIST,PMODIFY,PREVIEW,PSHIP,                 +
   PUTILITY,RETRIEVE,SIGNIN,SIGNOVR,UPDATE,VALIDATE)
FUNCEQU SAFAUTH=ALTER,                                   +
         C1ACTNS=(ALTER)
FUNCEQU TYPE=END
```

To change the mapping of access level to authorization values for ACTION_INITIATION or PACKAGE_ACTIONS, code the FUNCEQU entry according to the following syntax:

```
FUNCEQU SAFAUTH=(auth),
C1ACTNS=(c1access,c1access,...,c1access)
```

auth

The SAF authorization value equated with the access level (*c1access*). Valid *auth* values follow:

- NONE
- READ
- UPDATE
- CONTROL
- ALTER

c1access

The access level equated with the SAF authorization value (*auth*). Valid *c1access* levels are:

- ADD
- ALTER
- ARCHIVE
- DELETE
- DISPLAY
- ENVRNMGR
- GENERATE
- MOVE
- PBACKOUT
- PCAST
- PCOMMIT
- PCREATE
- PDISPLAY
- PDYNAMIC
- PEXECUTE
- PLIST
- PMODIFY
- PREVIEW
- PUTILITY
- PSHIP
- RETRIEVE

- SIGNIN
- SIGNOVR
- UPDATE
- VALIDATE

Important! You can associate each c1access value with only one auth value. For example, you cannot associate the GENERATE c1access value to both READ and CONTROL auth values.

If you code an *auth* value of NONE, a security check (a RACROUTE request) is not issued for the *c1access* functions it covers. Omitting C1ACTN from the definitions results in a default assignment of NONE. Use NONE when a security check is not desired. For example, the following code results in no security with DISPLAY and RETRIEVE.

```
FUNCEQU SAFAUTH=(NONE),                                          X
        C1ACTNS=(DISPLAY,RETRIEVE)
```

Map Authorization Values

CA ACF2 for z/OS, and CA Top Secret for z/OS users should be aware that a secondary mapping of the RACROUTE authorization value to CA Top Secret for z/OS or CA ACF2 for z/OS equivalents occurs in the SAF interface supplied with CA ACF2 for z/OS or CA Top Secret for z/OS.

The following are the map authorization values:

SAF Value	RACF Values	CA ACF2 for z/OS Value	CA Top Secret for z/OS Value
Read	Read	Read	Read
Update	Update	Write	Update
Control	Control	Write	Control
Alter	Alter	Allocate	Control

Note: If you specify a value other than READ you may need to update the SAF authorization when a new release of your site security package alters mapping values.

The required access level is determined at the action initiation security control point. You should be aware that CA ACF2 for z/OS, and CA Top Secret for z/OS downgrade the control authority to UPDATE or WRITE for non-VSAM data sets. You must take this into account when setting up ESI security rules.

Securing Actions Using the FUNCEQUE Entry

The FUNCEQU example provided in Defining SAF Authorization Levels shows a single level authorization. This means that all MENUAUTH/ACTIONS are defined in the FUNCEQU entry with a single SAF value or attribute level of READ. When a user requests an action, the pseudo data set is built and is passed to the site security package. The user's authorization level is returned to ESI and the user can invoke the action for as long as the user has READ access to the pseudo data set.

You can also secure actions by modifying the FUNCEQU entry. Actions should be logically grouped and mapped to different SAF values or attribute levels. When the ACTION_ INITIATION or PACKAGE_ACTIONS security control point is encountered, the pseudo data set is built and passed to the site security package. The user's authorization level is returned to ESI and is compared to the SAF value coded for the action. If the user's authorization level is equal to or greater than the level defined in the SAFAUTH parameter the action is allowed. Otherwise, the action is denied.

The following is an example of a modified FUNCEQU entry with four authorization levels:

```
FUNCEQU SAFAUTH=READ,                                          X
   C1ACTNS=(RETRIEVE,SIGNIN,PDISPLAY,PLIST)
FUNCEQU SAFAUTH=UPDATE,                                        X
   C1ACTNS=(ADD,UPDATE,GENERATE)
FUNCEQU SAFAUTH=CONTROL,                                       X
   C1ACTNS=(MOVE,SIGNOVR,ARCHIVE,DELETE)
FUNCEQU SAFAUTH=ALTER,                                         X
   C1ACTNS=(ENVRNMGR,ALTER,                                     X
   PCREATE,PCAST,PREVIEW,PEXECUTE, PDYNAMIC,                   X
   PBACKOUT,PCOMMIT,PSHIP,PUTILITY)
FUNCEQU TYPE=END
```

In order to initiate a MOVE, SIGNOUT OVERRIDE, ARCHIVE, or DELETE action, the user must have control authority to the pseudo data set. Because DISPLAY is not explicitly coded in a FUNCEQU entry and therefore defaults to SAFAUTH=NONE, all users are granted DISPLAY access provided they pass other appropriate security control points.

How to Define SAF Name Formats

ESI determines data set name formats by checking the NAMEQU entry in the Name Equates Table. The following excerpt from the sample Name Equates Table shows the NAMEQU entries that create the ENTITY=dsname value used by the RACROUTE request.

```
     NAMEQU ENVIRONMENT_ACCESS,                    +
        L1=('C1'),                                 +
        L2=('ENVIRON'),                            +
        L3=(ENVIRONMENT)
     NAMEQU PRIMARY_OPTIONS,                       +
        L1=('C1'),                                 +
        L2=(ENVIRONMENT),                          +
        L3=('PMENU'),                              +
        L4=(MENUITEM)
     NAMEQU FOREGROUND_OPTIONS,                    +
        L1=('C1'),                                 +
        L2=(ENVIRONMENT),                          +
        L3=('FORACTN'),                            +
        L4=(MENUITEM)
     NAMEQU ACTION_INITIATION,                     +
        L1=('C1'),                                 +
        L2=(ENVIRONMENT),                          +
        L3=(SYSTEM),                               +
        L4=(SUBSYSTEM)
     NAMEQU ACTION_INITIATION,                     +
        L1=('C1'),                                 +
        L2=(MENUAUTH)
     NAMEQU PACKAGE_ACTIONS,                       +
        L1=('C1'),                                 +
        L2=('PACKAGE'),                            +
        L3=(MENUITEM),                             +
        L4=(PKGSUBFC),                             +
        L5=(PKGID)
    NAMEQU CONCURRENT_ACT_PROC,                    +
        CLASS='DATASET',                           +
        WARN=NO,                                   +
        LOG=NONE,                                  +
        L1=('C1'),                                 +
        L2=('CAP')
```

Each format (ENVIRONMENT_ACCESS through CONCURRENT_ACT_PROC) is defined only once within the Name Equates Table. (An optional second format is allowed for ACTION_INITIATION.)

Many installations have unique naming standards and index levels for data set names. ESI allows you to customize data set names to conform to your site's conventions. In addition, you can establish your own security levels for any given field.

Note: These names only represent data set access rules. There are no physical data sets associated with these rules.

Change Names Generated at Security Control Points

To change the names generated at each security point, code the NAMEQU entry according to the following syntax:

```
NAMEQU FORMAT,                                                  X
       Ln=(field1(begin,length),...fieldn(begin,length)),       X
       CLASS=classname,                                         X
       LOG=(ASIS|NONE|NOFAIL|NOSTAT),                           X
       WARN=(YES|NO)
NAMEQU TYPE=END
```

security call FORMAT

The data set name format (ENVIRONMENT_ACCESS through CONCURRENT_ACT_PROC)

L*n*

The index level of the data set name. Data set names are generated in the form of: L1.L2.L3.L4. Note that the index levels generated are separated from other levels by a period (.). *n* specifies a value from 1-10. The *fieldn* values described next specify the index levels.

***fieldn* (begin, length)**

A literal or a CA Software Change Manager keyword which is placed into the specified index level. *Fieldn* can be any literal of up to eight characters. Note that literals must be enclosed in single quotation marks. *(begin, length)* is an optional parameter which allows you to specify a portion of a CA Endevor SCM keyword.

Note: For more information about the *begin* and *length* parameters, see How to Specify a Substring of a Keyword (see page 71).

classname

The literal DATA SET or a user-defined resource class. DATA SET is the default if *classname* is not coded. A *classname* value other than DATASET may yield unpredictable results such as shortening the data set name.

Note: For more information, see How to Define a Class Other Than Data Set with RACF (see page 77).

LOG=(ASIS| NONE| NOFAIL|NOSTAT)

Specifies whether the security software at your site logs access attempts. The LOG parameter is used on the RACROUTE macro. SMF records written by site security as a result of the LOG parameter are in addition to, and separate from, CA Endevor SCM written SMF records. For more information about CA Endevor SCM SMF logging, see The SMF Interface in the Administration Guide. The valid values are:

- *ASIS*—Records access attempts as specified by the operating system ADDSD and ALTDSD operator commands, or with the RDEFINE and RALTER commands for tape or DASD volumes (if the CLASS= parameter specifies something other than data set). ASIS gives the security package control over what is logged based on the profile AUDIT options, the user UAUDIT attribute, and SETROPTS LOGOPTIONS settings.
- *NONE*—Suppresses logging by site security. If you are a CA ACF2 for z/OS user, set this parameter. This is the default.
- *NOFAIL*—Records access attempts depending on authorization check results. Does not record an access attempt, if the authorization check fails. If the authorization check succeeds, the access attempt is recorded. When used with a site security AUDIT setting of FAILURES(READ), the site security does not log a successful authorization check
- *NOSTAT*—The access attempt is not recorded and resource statistics are not updated.

WARN= *(YES|NO)*

Specifies the local warning option. This parameter overrides the ESIDFLTS value. The YES option turns on warning mode for the resource name (the security control point). The NO option turns off warning mode. If *YES* or *NO* is not specified, then the action defaults to the coding specified in the ESIDFLTS WARN option is not specified, the default value is *WARN=NO*.

Note: For more information, see The ESI Warning Mode (see page 109).

TYPE=END

Indicates the end of the name equates entries and can be specified separately or on the last NAMEQU FORMAT*n* entry.

How to Specify a Substring of a Keyword

You can specify a substring of a keyword, according to the following syntax:

Ln=(KEYWORD(B,L))

B

The first character of the keyword (beginning column relative to one).

L

The number of characters to extract from the keyword.

For example, to obtain a field value of FIN to represent a system named FINANCE, you would code the following:

L1=(SYSTEM(1,3))

You can concatenate field values by specifying more than one field, each separated by a comma. For example, to obtain a field value of ENVIPROD to represent an environment named PRODUCTION, you would code the following:

L2=('ENVI',ENVIRONMENT(1,4))

Note: When data set names are generated, all trailing and embedded blanks are compressed. A value of '$' occurs in the index level of the name if the resulting index level is all blanks (that is, not available).

The following lists the keywords and the formats for which the keywords are available.

Keyword	ENV	PRI	FOR	ACTS	PKG	CAP
ENVIRONMENT	X	X	X	X		
ACTION		X	X	X	X	
MENUITEM		X	X	X	X	
MENUAUTH		X	X	X	X	
SYSTEM				X		
SUBSYSTEM				X		
STAGEID				X		
STAGENAME				X		
STAGENO				X		
TYPE				X		
ELEMENT				X		

Keyword	ENV	PRI	FOR	ACTS	PKG	CAP
ELM-10				X		
CCID				X		
PKGSUBFC					X	
PKGID					X	
PKGTYPE					X	
PKGSTAT					X	
PKGAPPGR					X	
PKGBOE					X	
PKGSHR					X	
PKGPROM					X	

Note: The format names used in the table have been abbreviated as: ENVIRONMENT_ACCESS=ENV, PRIMARY_OPTIONS=PRI, FOREGROUND_OPTIONS=FOR, ACTION_INITIATION=ACTS, PACKAGE_ACTIONS=PKG and CONCURRENT_ACT_PROC=CAP.

A brief definition of each keyword follows:

ENVIRONMENT (1)

The CA Endevor SCM environment name

ACTION

The CA Endevor SCM access level for which a request is made. The ACTION keyword is interchangeable with the MENUAUTH keyword. The following values are valid for action: ADD, ALTER, ARCHIVE, DELETE, DISPLAY, ENVRNMGR, GENERATE, MOVE, PBACKOUT, PCAST, PCOMMIT, PCREATE, PDISPLAY, PEXECUTE, PLIST, PMODIFY, PREVIEW, PUTILITY, RETRIEVE, SIGNIN, SIGNOVR, UPDATE, VALIDATE.

For more information about how these access levels relate to the action being performed, see The Default Authorization Value (see page 86).

MENUITEM

Allows individual line tailoring of the Primary Options menu, the Foreground Options menu, and the Package Actions menu.

- When used with *PRIMARY_OPTIONS* (Primary Options menu), the following values are valid: DISPLAY, FOREGRND, BATCH, PACKAGE, BATCHPKG, USER, ENVRMENT, LOAD, UNLOAD, RELOAD.

  Note: For more information about these values, see The Primary Options Security Control Point (see page 34).

- When used with *FOREGROUND_OPTIONS* (Foreground Options menu), the following values are valid: DISPLAY, ADDUPDT, RETRIEVE, GENERATE, MOVE, DELETE, PRINT, SIGNIN.

  Note: For more information about these values, see The Foreground Options Security Control Point (see page 83).

- When used with ACTION_INITIATION, the name of the action being performed. The following values are valid: ADD, ALTER, UPDATE, DISPLAY, RETRIEVE, GENERATE, MOVE, DELETE, PRINT,RESTORE, SIGNIN, TRANSFER LIST, ARCHIVE.

  Note: For more information about these values, see The Action Initiation Security Control Point (Standard) (see page 85).

- When used with *PACKAGE_ACTIONS*, the following values are valid: BACKOUT, CAST, COMMIT, CREATE, DISPLAY, EXECUTE, MODIFY, REVIEW, UTILITY, DYNAMIC.

  Note: For more information about these values, see The Package Action Security Control Point (Extension) (see page 88).

SYSTEM (1)

The CA Endevor SCM system name

SUBSYSTEM (1)

The CA Endevor SCM subsystem name

STAGEID

The CA Endevor SCM stage ID (1 character value) is taken from the STG1= and STG2= parameters in the C1DEFLTS TYPE=ENVIRONMENT.

STAGENAME (1)

The CA Endevor SCM stage name

TYPE (1)

The CA Endevor SCM element type

ELEMENT

The first eight characters of the element name

ELEM-10

The 10 characters of the element name

CCID

The current change control identifier. A CCID can be up to 12 characters. If an SCL action statement for add, archive, delete, generate, move, restore, retrieve, transfer, or update specifies a CCID, that *ccid* is used. If no CCID is specified, $ is substituted for the CCID field. If the action does not use a CCID (for example, display, signin, list, copy, or print), $ is substituted for the CCID field.

PKGSUBFC (2)

The sub-menu function code in which the following values are valid:

- ACTSUMM
- ADD
- APPROVE
- APPROVER
- BACKIN
- BACKOUT
- BUILD
- CAST
- COMMIT
- CONFIRM
- COPY
- CORRINFO
- DELETE
- DENY
- EABKO

  Specifies Element Action Backout.
- EABKI

  Specifies Element Action Backin.
- EDIT
- EXECUTE
- EXPORT
- IMPORT

- LIST
- PACKAGE
- REPORTS
- RESET
- SCL
- STAGE
- UPDATE
- XMIT

PKGID (3)

The 1-to-16-character user-defined package name

PKGTYPE (3, 4)

Defines whether the package is emergency or standard:

- STANDARD
- EMERGENCY

PKGSTAT (3, 4)

The package status:

- IN-EDIT
- IN-APPROVAL
- DENIED
- APPROVED
- IN-EXECUTION
- EXECUTED
- EXEC-FAILED
- COMMITTED

PKGAPPGR

The approver group name

PKGBOE

The backout-enabled status of the package:

- Y
- N

PKGSHR

The share option associated with the package:

- Y
- N

PKGPROM

Promotion package indicator:

- Y
- N

Note: For the package symbolics, a value of '$' is substituted for the variable when the real value is not available to CA Endevor SCM. For example, prior to a CAST, the value for PKGAPPGR is a $.

(1) Up to eight characters in length.

(2) The value for PKGSUBFC always resolves to LIST when selecting an option from the package Foreground Options menu. The values listed above are only available from the corresponding foreground panel. For example, PKGSUBFC is resolved to LIST when selecting option 3 from the package Foreground Options menu and CAST from the Cast panel.

(3) Each node in a data set name only allows a maximum of eight characters per value. PKGIDs can be up to 16 characters in length. Use substringing to limit the number of characters in the generated node, otherwise, security failures could result from what your security package may consider an invalid data set name. For example, coding pkgid(1,8) will substitute VERYLONG for pkgid VERYLONGPKGIDNAME.

(4) PKGSTAT and PKGTYPE values can be more than eight characters (z/OS only supports eight characters in a data set name node). By default, only the first 8 characters of these fields are used (for example, IN-APPROVAL truncates to IN-APPRO, EMERGENCY truncates to EMERGENC). Make sure to follow these conventions when defining these profiles in your security package.

Important! If the pseudo data set name is greater than 44 characters, it is automatically shortened.

Define a Class Other Than Data Set with RACF

The RACROUTE macro has a default value of DATASET for the CLASS parameter. Depending on the granularity of your site security rules, you could have many rules that belong in this category. You can reduce the number of rules that are searched during authorization by creating an additional class category that is specific to ESI security rules. In this section, the resource class is called $ENDEVOR.

Important: For the creation of resource classes, consult with your Security administrator and, if necessary, IBM. The appropriate updates to your site's IBM RACF class description table (CDT) depend on your site's particular security software configuration. The configuration can be unrelated to CA Endevor SCM and therefor outside the scope of the CA Endevor SCM Support team's knowledge. For more information about the parameters, see IBM's *RACF Macros Interface Manual*.

To define a class other than DATASET

1. Add the resource class name $ENDEVOR to the RACF class description table (CDT).

   Use the following macro to create a class called $ENDEVOR.

```
 $ENDEVOR ICHERCDE  CLASS=$ENDEVOR,                                    X
           id=141,                                                     X
           MAXLNTH=246,                                                X
           FIRST=ALPHANUM,                                             X
           OTHER=ANY,                                                  X
           POSIT=20,                                                   X
           RACLIST=ALLOWED,                                            X
           DFTUACC=NONE,                                               X
           OPER=NO
 ICHRRCDE ICHERCDE
     END
/*
//*
//INSTL3  EXEC IPOSMPF, COND=(0,NE),
//          CSI='MVSSMPF.GLOBAL.CSI'
//SMPEIN  DD *
SET BDY(GLOBAL).
REJECT S(NRACCDT)BYPASS(APPLYCHECK).
RECEIVE S(NRACCDT) SYSMODS.
SET BDY(M220TAF).
```

   Note: ESI supports resource names up to 246 bytes long. In order for you to use extended resource names, an installation-defined Class must be used. In addition, you must insure that your site's security package supports extended names. For more information, see your security package documentation.

2. This step is *not* applicable for z/OS 1.6 and above. Add the entry in the following example to the RACF router table. The processor below executes these two steps.

```
TAB16   ICHRFRTB   CLASS=$ENDEVOR,ACTION=RACF
TABEND   ICHRFRTB   TYPE=END
      END ICHRFR01
/*
//*
//INSTL3 EXEC IPOSMPF,COND=(4,NE),
//        CSI='MVSSMPF.GLOBAL.,CSI'
//SMPEIN  DD *
 SET BDY(GLOBAL)      B .
 REJECT S(NRACRTT)      BYPASS(APPLYCHECK)  .
 RECEIVE S(NRACRTT)      SSMODS .
 SET DBY (M220TAF       .
 APPLY S(NRACRTT)       REDO
/*
//SMPPTFIN DD       DSN=&&LOADSET.,DISP=(OLD,DELETE)
```

3. Assemble the CDT and the router table and IPL the system.

4. Activate the $ENDEVOR resource class using the RACF SETROPTS command. This activates generic access checking for the resource class.

5. Modify the Name Equates Table. The following shows the changes that should be made to the NAMEQU entries.

```
NAMEQU ENVIRONMENT_ACCESS,                                    X
       L1=('C1'),                                             X
       L2=('ENVIRON'),                                        X
       L3=(ENVIRNOMENT),                                      X
       CLASS='$ENDEVOR'

NAMEQU PRIMARY_OPTIONS,                                       X
       L1=('C1'),                                             X
       L2=('PMENU'),                                          X
       L3=(ENVIRONMENT),                                      X
       L4=(MENUITEM),                                         X
       CLASS='$ENDEVOR'

NAMEQU FOREGROUND_OPTIONS,                                    X
       L1=('C1'),                                             X
       L2=('FMENU'),                                          X
       L3=(ENVIRONMENT),                                      X
       L4=(MENUITEM),                                         X
       CLASS='$ENDEVOR'
```

```
NAMEQU ACTION_INITIATION,                                    X
       L1=('C1'),                                            X
       L2=(ENVIRONMENT),                                     X
       L3=(SYSTEM),                                          X
       L4=(SUBSYSTEM),                                       X
       L5=(MENUAUTH),                                        X
       CLASS='$ENDEVOR'

NAMEQU PACKAGE_ACTIONS,                                      X
       L1=('C1'),                                            X
       L2=('PACKAGE'),                                       X
       L3=(MENUITEM),                                        X
       L4=(PKGSUBFC),                                        X
       L5=(PKGID),
       CLASS='$ENDEVOR'

NAMEQU CONCURRENT_ACT_PROC,                                  X
       L1=('C1'),                                            X
       L2=('CAP')                                            X
      CLASS='$ENDEVOR'                                       X
```

Coordinating Access Levels, Menu Options, and Authorization Levels Using the RACROUTE Request

The RACROUTE request is derived from rules you define in the Name Equates Table. Formats ENVIRONMENT_ACCESS through CONCURRENT_ACT_PROC coordinate access levels, menu options, and authorization levels with the site security packages (RACF, CA ACF2 for z/OS, and CA Top Secret for z/OS).

Formats are defined in the Name Equates Table, a portion of which is shown in the following example:

```
NAMEQU ENVIRONMENT_ACCESS,                                   +
   L1=('C1'),                                                +
   L2=('ENVIRON'),                                           +
   L3=(ENVIRONMENT)
NAMEQU PRIMARY_OPTIONS,                                      +
   L1=('C1'),                                                +
   L2=(ENVIRONMENT),                                         +
   L3=('PMENU'),                                             +
   L4=(MENUITEM)

NAMEQU FOREGROUND_OPTIONS,                                   +
   L1=('C1'),                                                +
   L2=(ENVIRONMENT),                                         +
   L3=('FORACTN'),                                           +
   L4=(MENUITEM)
```

```
NAMEQU ACTION_INITIATION,                    +
   L1=('C1'),                                +
   L2=(ENVIRONMENT),                         +
   L3=(SYSTEM),                              +
   L4=(SUBSYSTEM)

NAMEQU ACTION_INITIATION,                    +
   L1=('C1'),                                +
   L2=(MENUAUTH)

NAMEQU PACKAGE_ACTIONS,                      +
   L1=('C1'),                                +
   L2=('PACKAGE'),                           +
   L3=(MENUITEM),                            +
   L4=(PKGSUBFC),                            +
   L5=(PKGID)

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

Under each format entry, rules define parts of every RACROUTE request. The format defines how the pseudo data set is built. Variables in each rule appear without single quotes and literals appear within single quotes. CA Endevor SCM substitutes the appropriate value for each variable.

1. Environments (*ENVIRONMENT_ACCESS*): Restricts user access to environments. ENVIRONMENT_ACCESS calls are issued during CA Endevor SCM initialization both in foreground and batch.
2. Primary Options panel (PRIMARY_OPTIONS): The primary options panel is customized for each user, based on the rules set up by the security administrator. Only options that the user can select are displayed on the Primary Options panel. PRIMARY_OPTIONS calls are issued before the Primary Options panel is displayed.
3. Foreground Options panel (FOREGROUND_OPTIONS): Only options that the user can select are displayed on the Foreground Options panel. FOREGROUND_OPTIONS calls are issued before the Foreground Options panel is displayed.
4. Action initiation (ACTION_INITIATION): Prior to performing a selected action, ESI requests a ruling to determine whether the user has the authorization to perform the action. ESI makes a pass for each defined ACTION_INITIATION format.

5. Action initiation (ACTION_INITIATION): The extension ACTION_INITIATION is an optional extension of the standard ACTION_INITIATION to allow for names longer than 44 bytes. This format is called only if the standard ACTION_INITIATION is successful.
6. Package actions (PACKAGE_ACTIONS): Prior to performing an action against a package, ESI requests a ruling to determine whether the user has the authorization to perform that action against the package.
7. Concurrent Action Processing (CONCURRENT_ACT_PROC): Prior to initiating (spawning) concurrent batch actions, ESI requests a ruling to determine whether the user has the authorization to use this facility. If the requestor does not have CAP access, an error message is issued and processing is terminated.

The Environment Access Security Control Point

The Environment access security control point occurs at the following places:

- During CA Endevor SCM initialization, prior to display of the Environment Selection menu in the foreground and prior to processing of any actions in batch.
- During LOAD utility operations to verify the user's access to the desired inventory location.
- During UNLOAD and RELOAD operations to re-verify the user's authority to backup and restore the desired inventory locations.

Note: When the user changes environments in the foreground, CA Endevor SCM looks up in the user's accessible environments to see whether you have access to the environment. No ESI call is issued during this processing because it is not necessary.

Use the Environment Selection menu to select an environment. Any other panel that displays the ENVIRONMENT field allows you to switch environments.

When CA Endevor SCM starts under ISPF, CA Endevor SCM ESI issues a RACROUTE request using ENVIRONMENT_ACCESS for each environment defined in the C1DEFLTS Table. Every authorized environment is then displayed on your Environment Selection menu.

Define rules for your site security package based on the ENVIRONMENT_ACCESS resource names. These rules determine which environments are accessible to the user and are displayed on the Environment Selection menu. You must have READ authority for each resource to gain access to the specified environment.

Example: ENVIRONMENT_ACCESS rules

This becomes: C1.ENVIRON.*environment*, where *environment* is the name for which access is requested. If you have access to only one environment, the Environment Selection menu is not presented. If you do not have access to any environments, CA Endevor SCM is not available.

```
NAMEQU ENVIRONMENT_ACCESS,                    X
       L1=('C1'),                             X
       L2=('ENVIRON'),                        X
       L3=(ENVIRONMENT)
```

Note: If you use environment mapping you must also have access to all forward environments up the map.

The security administrator at a site with two environments (QA and PROD) wants to give a programmer access to both environments. To do this, they must define a data set access rule for the site security package that gives the programmer READ access to the data set names:

C1.ENVIRON.QA

C1.ENVIRON.PROD

Primary Options Security Control Point

The Primary Options security control point occurs prior to building the Primary Options menu for the current environment in foreground, after access is granted through the Environment Selection menu or during LOAD/UNLOAD/RELOAD.

Define rules for your site security package based on the PRIMARY_OPTIONS data set names to determine the option(s) to be displayed on the user's Primary Options menu. A user must have READ authority for each data set name to gain access to the specified primary option.

A sample of PRIMARY_OPTIONS rules is shown next:

```
NAMEQU PRIMARY_OPTIONS,
       L1=('C1'),
       L3=('PMENU'),
       L2=(ENVIRONMENT),
       L4=(MENUITEM)
```

This becomes: C1.*environment*.PMENU.*menuitem*

environment

The environment you are trying to access

menuitem

The corresponding Menu Item value.

The Foreground Options Security Control Point

The Foreground Options security control point occurs prior to building the Foreground Options menu for the current environment in foreground, after access is granted through the Primary Options menu.

Define rules for your site security package based on the FOREGROUND_OPTIONS data set names to determine the options to be displayed on the user's Foreground Options menu. A user must have READ authority for each data set name to gain access to the specified foreground option.

The following is a sample of FOREGROUND_OPTIONS rules:

```
NAMEQU FOREGROUND_OPTIONS,
       L1=('C1'),
       L2=(ENVIRONMENT),
       L3=('FORACTN'),
       L4=(MENUITEM)
```

This becomes: C1.*environment*.FORACTN.*menuitem*

environment

The environment you are trying to access

menuitem

The corresponding Menu Item value.

The following lists the value and how the menu item is displayed:

- DISPLAY
- ADDUPDT (Item displayed as ADD/UPDATE)
- RETRIEVE
- GENERATE
- MOVE
- DELETE
- PRINT
- SIGNIN

Note: This security point can only be reached if a PRIMARY_OPTIONS rule allows access to the Foreground Actions menu for the current environment.

FOREGROUND_OPTIONS rules do not apply to batch operations.

Example

The security administrator at a site with an environment called QA wants to give a programmer access to the ADD/UPDATE, RETRIEVE, PRINT, and SIGNIN options. To do this, you must define a data set access rule for the site security package (RACF, CA ACF2 for z/OS, and CA Top Secret for z/OS) that gives the programmer READ access to the data sets:

- C1.QA.FORACTN.ADDUPDT
- C1.QA.FORACTN.RETRIEVE
- C1.QA.FORACTN.PRINT
- C1.QA.FORACTN.SIGNIN

The Action Initiation Security Control Point (Standard)

The Action Initiation security control point occurs:

- Prior to performing a CA Endevor SCM action.
- Prior to a cast operation during package processing, for each action in a package, if the PKGCSEC flag is set to **Y**.
- Prior to an inspect operation during package processing, for each action in a package, if the PKGISEC flag is set to **Y**.
- During package verification processing.

Define rules for the site security package based on the ACTION_INITIATION data set names. These rules determine the CA Endevor SCM actions the user can perform. A user must have the proper level of authority to each data set name (based on action) to gain access to the specified CA Endevor SCM action.

You need to write rules to secure the source and target locations for ACTION_INITIATION.

Sample ACTION_INITIATION rules are shown next.

```
NAMEQU ACTION_INITIATION,
       L1=('C1'),
       L2=(ENVIRONMENT),
       L3=(SYSTEM),
       L4=(SUBSYSTEM)
```

This becomes: C1.*environment.system.subsystem*

environment

The environment you are trying to access

system

The system you are trying to access

subsystem

The subsystem you are trying to access

The Action Initiation Security Control Point (Extension)

The extension ACTION_INITIATION is an optional extension of the standard ACTION_INITIATION that can allow for names longer than the class attribute allows. Access to both ACTION_INITIATIONs is required and you must write rules to secure the source and target locations for them.

Note: This is only called if ACTION_INITIATION has RC=0.

Sample extension ACTION_INITIATION rules are shown next.

```
NAMEQU ACTION_INITIATION,
       L1=('C1'),
       L2=(MENUAUTH)
```

This becomes: C1.*menuauth*

Where *menuauth* is the access level required for the requested action.

Example

This example demonstrates how both ACTION_INITIATIONs rules work together. Assume you are a security administrator and you need to define data set access rules for your site security package. You have an environment called QA and you want to give a programmer access to a system called FINANCE and a subsystem called ACTSPAY. In addition, you want to limit the actions the programmer can perform in the subsystem ACTSPAY to RETRIEVE and DISPLAY. To accomplish this you must write both ACTION_INITIATIONs rules. These rules grant the programmer READ access to these pseudo data sets:

- C1.QA.FINANCE.ACTSPAY (ACTION_INITIATION)
- C1.RETRIEVE (ACTION_INITIATION)

The Default Authorization Value

The *authorization* value for each RACROUTE request is translated in the following table. The table reflects the delivered sample BC1TNEQU table that is described in the Name Equates Table.

Note: If no FUNCEQU macro entry is specified for an action, it will default to SAFAUTH=NONE, meaning that there will be no security call for this action.

When you determine the authorization access for a user, be sure to check the "Access level required" column to ensure that you are not giving users access to activities that you do not want them to access.

To Perform this Activity	Access Level Required	SAF Authorization Level
Add	Add	READ
Update	Update	READ
Retrieve	Retrieve	READ
Generate (Stage 1 or Entry stage)	Generate	READ
Generate (Stage 2 non-entry	Move	READ
Move (from Stage 1 or Entry Stage) (1)	Move	READ

To Perform this Activity	Access Level Required	SAF Authorization Level
Move (from Stage 2 non-entry) (1)	Move	READ
Move (to Stage 1 or Entry Stage)	Add	READ
Move (to Stage 2 non-entry)	Move	READ
Display from selection list	Display	READ
Browse	Retrieve	READ
Delete (Stage 1 or Entry Stage)	Delete	READ
Delete (Stage 2 non-entry)	Move	READ
Signin	Signin	READ
Print	Display	READ
Transfer (from Stage 1 or Entry Stage)	Add/Update	READ
Transfer (to Stage 2 non-entry)	Move	READ
Transfer (from Stage 1 or Entry Stage with Delete)	Delete	READ
Transfer (from Stage 2 non-entry with Delete)	Move	READ
Transfer (from Stage 1 or Entry Stage without Delete)	Retrieve	READ
Transfer (from Stage 2 non-entry without Delete)	Retrieve	READ
Alter	Alter	ALTER
Archive	Archive	READ
Restore to Stage 1 or Entry Stage	Add	READ
Restore to Stage 2 non-entry	Move	READ
Copy	None	READ
List	Display	READ
Signout Override (2)	Signovr	READ
Validate	Display	READ
Actions against any elements of type processor (3)	Envrnmgr	READ

(1) If in-house security is designed to expect the move action to issue a security call at the target location, then do a security check at the target location during the move processing. To do this you must activate ENHOPT SEC_MOVE_TARGET=ON in the ENCOPTB

(2) Signout override is always the second call in an action. The first call is the specific action involved - Add or Delete, for example - and then, if necessary, the call for signout overrid is performed.

(3) When performing actions against type processors, two calls are issued. The first call checks whether the user can perform the specific action involved, such as Add or Delete. If the user is able to perform that action, the second call is issued for access level ENVRNMGR to see if the user also has permission to work with type processors.

The Package Actions Security Control Point

The Package Actions security control point occurs prior to performing a package action.

Define rules for the site security package based on the PACKAGE_ACTIONS data set names. These rules determine the CA Endevor SCM actions the user can perform. You must have the proper level of authority to each data set name (based on action) to gain access to the specified CA Endevor SCM action.

You need to write rules to secure the source and target locations for PACKAGE_ACTIONS.

The following are sample PACKAGE_ACTIONS rules:

```
NAMEQU PACKAGE_ACTIONS,
       L1=('C1'),
       L2=('PACKAGE'),
       L3=(MENUITEM),
       L4=(PKGSUBFC),
       L5=(PKGID)
```

This becomes: C1.PACKAGE.*menuitem.pkgsubfc.pkgid*

menuitem

Allows individual line tailoring of the Primary Options menu. The following values are valid:

- BACKOUT
- CAST
- COMMIT
- CREATE
- DISPLAY
- EXECUTE

- MODIFY
- REVIEW
- SHIP
- UTILITY

Pkgsubfc

The sub-menu function code or action (for example, CREATE allows you to build, import, export, and so forth). One of the following values:

- ACTSUMM
- ADD
- APPROVE
- APPROVER
- BACKIN
- BACKOUT
- BUILD
- CAST
- COMMIT
- CONFIRM
- COPY
- CORRINFO
- DELETE
- DENY
- EABKO

  Specifies Element Action Backout.
- EABKI

  Specifies Element Action Backin.
- EDIT
- EXECUTE
- EXPORT
- IMPORT
- LIST
- PACKAGE

- REPORTS
- RESET
- SCL
- STAGE
- UPDATE
- XMIT

The value for *pkgsubfc* always resolves to LIST when selecting an option from the package Foreground Options menu. The values listed above are only available from the corresponding foreground panel. For example, menuitem is resolved to LIST when selecting option 3 from the package Foreground Options menu and CAST from the Cast panel.

pkgid

The user-defined package name

The NAMEQU macro supports the FORMAT=PACKAGE_ACTION entry along with the new symbolics required to build the model name. The format of the model name is variable, but can include the following specific package symbolics:

PKGSHR

The share option associated with the package:

- Y
- N

PKGPROM

Promotion package indicator:

- Y
- N

PKGBOE

The backout-enabled status of the package:

- Y
- N

PKGAPPGR

Approver group name

PKGSTAT

The package status:

- IN-EDIT
- IN-APPROVAL

- DENIED
- APPROVED
- IN-EXECUTION
- EXEC-FAILED
- EXECUTED
- COMMITTED

Status names can be more than eight characters long (z/OS only supports eight characters in a data set name node). Use substringing to limit the number of characters in the generated node. By default, the value is the first eight characters of the status name (for example, IN-APPROVAL shortens to IN-APPRO).

PKGTYPE

Defines whether the package is emergency or standard:

- STANDARD
- EMERGENCY

Each node in a data set name only allows a maximum of eight characters per value; therefore, values greater than eight characters are truncated to eight characters (for example, EMERGENC).

PACKAGE_ACTIONS ESI Calls

The following table displays some of the rules that are built for this sample. The pseudo data set names varies depending on how your PACKAGE_ACTIONS NAMEQU is configured.

Menuitem/ Subfunction	FUNCEQU Name	Sample Pseudo Data Set Security Rule C1.PACKAGE.*menuitem.pkgsubfc.pkgid*
BACKOUT	PLIST	C1.PACKAGE.DISPLAY.LIST.PKG001
Backout	PBACKOUT	C1.PACKAGE.BACKOUT.BACKOUT.PKG001
Display	PDISPLAY	C1.PACKAGE.DISPLAY.BACKOUT.PKG001
Backin	PBACKOUT	C1.PACKAGE.BACKOUT.BACKIN.PKG001
CAST	PLIST	C1.PACKAGE.DISPLAY.LIST.PKG001
Cast	PCAST	C1.PACKAGE.CAST.CAST.PKG001
SCL	PDISPLAY	C1.PACKAGE.DISPLAY.SCL.PKG001

Note: There is no security for the Notes subfunction.

Concurrent Action Processing Security Control Point

The Concurrent Action Processing Security Control Point occurs before the dispatch of the first element action that is eligible for concurrent processing. It only occurs if the user has requested Concurrent Action Processing.

If concurrent action processing is requested for an batch job, a batch package execution or for an API execution, the ESI issues a RACROUTE request using CONCURRENT_ACT_PROC to determine if the user has access to this facility.

Define a rule for your site security package based on the CONCURRENT_ACT_PROC data set name. This rule determines if the concurrent action processing facility is accessible to the user. A user must have READ authority for the data set name to be able to request CAP processing.

Example: CONCURRENT_ACT_PROC rule

A sample concurrent action processing rule is shown next:

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

This rule becomes: C1.CAP

How to Enable ESI

This section explains how to enable CA Endevor SCM ESI, which is supplied on the installation tape. The procedure for enabling ESI should be performed after the product is installed and verified.

Follow this process to enable ESI and ensure your success:

1. Prepare your security worksheet.
2. Define rules for your site security package.
3. Customize the Name Equates Table (Formats ENVIRONMENT_ACCESS through CONCURRENT_ACT_PROC).
4. Assemble and link-edit the Name Equates Table (BC1TNEQU).
5. Assemble and link the Defaults Table (C1DEFLTS).
6. Test ESI security using warning mode.

The sections that follow explain each of these steps in greater detail.

How to Prepare Your ESI Security Worksheet

To begin planning your ESI security strategy, you must gather information about all the users and applications for which you are responsible. You need to determine how users and applications relate to each other and how detailed you want your security strategy to be.

We recommend using an ESI worksheet to help you define and categorize the various levels of access required by anyone involved in application development. Once you have completed your ESI worksheet, you can begin defining access levels.

Note: Blank worksheets for defining security rules are provided in Security Worksheets.

Keep an updated version of your ESI worksheet available for reference purposes. If you make changes to your Names Equates table or your site security package rules, remember to update your ESI worksheet as well. Use the worksheet to:

- Associate authorization levels with personnel or departments.
- List activities to include for selected departments in Stages 1 and 2 of each environment.
- Refer to when updating or revising your security rules.

How to Define Security Rules for Your Site

The following section contains four sample procedures that describe the steps you need to follow in order to determine security rules for your site. The procedures describe steps for defining the following security rules:

- Site environment rules
- Primary Options panel rules
- Foreground action rules
- Action initiation rules
- Package action rules
- Concurrent action processing rule

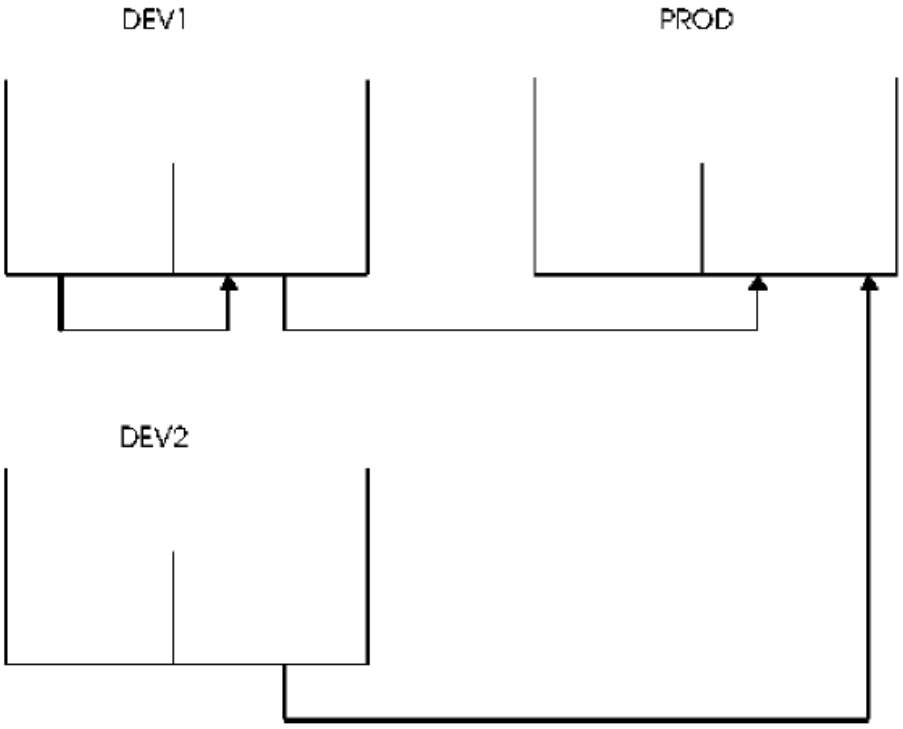
Note: Blank worksheets for defining security rules are provided in Security Worksheets.

Define Security Rules for Your Site Environments

The following four steps explain how to set up security for the environments at your site.

1. Plan a diagram of the flow for your site environments and label each box with the name of your environments.

The following example shows a site where parallel development occurs between environments DEV1 and DEV2.



2. Plan a table that lists your environments in the far left column and the users at your site along the top row. In the following example, an X indicates that a user has security access.

Envr	Appl Pgmr	Appl Mgr	Other Depts	QA	Prod Ctl	Tech Supp	Audit	Admin
DEV1	X	X				X		X
DEV2	X	X				X		X
PROD	X	X	X	X	X	X	X	X

3. Determine the format for your environment security rules. In order to generate a pseudo data set name in the format 'C1'.'ENVIRON'.*environment*, you must set up the NAMEQU entry in the Name Equates Table using this format:

```
NAMEQU ENVIRONMENT_ACCESS,
 L1=('C1'),
 L2=('ENVIRON'),
 L3=(environment)
```

4. In the next tables, determine the pseudo data set names based on the format described in Step 3 and the environments entered in Step 2.

Env	AP	AM	OD	QA	PC	TS	AU	AD	Data Set Names
DEV1	X	X				X		X	C1.ENVIRON.DEV1
DEV2	X	X				X		X	C1.ENVIRON.DEV2
PROD	X	X	X	X	X	X	X	X	C1.ENVIRON.PROD

Note: User access to data sets in indicated with an X. You must have read access to the data sets in order to have environment access.

User Abbreviations

These abbreviations are used in the security table's column heading to describe the users:

- AP - Application Programming
- AM - Application Management
- OD - Other Departments
- QA - Quality Assurance
- PC - Production Control
- TS - Technical Support
- AU - Audit
- AD - Administration

Define Security Rules for the Primary Options Panel

The following three steps explain how to set up security for the Primary Options Panel at your site.

1. Plan a table that lists the Primary Options panel menu items in the far left column and the users at your site along the top row. In the following example, an X indicates that a user has access to a menu item.

Menu Item	AP	AM	OD	QA	PC	TS	AU	AD
DISPLAY	X	X	X	X	X	X		X
FOREGROUND	X	X				X		X
BATCH	X	X				X		X
PACKAGE				X	X	X		X
USERMENU				X	X	X		X
ENVIRONMENT						X		X
LOAD (Batch action only)								X
UNLOAD (Batch action only)								X
RELOAD (Batch action only)								X
BATCH PACKAGE	X	X		X	X	X		X

2. Determine the format for your Primary Options panel security rules. In order to generate a pseudo data set name in the format 'C1'.environment.'PMENU'.*menuitem*, you must set up the NAMEQU entry in the Name Equates Table using this format:

```
NAMEQU PRIMARY_OPTIONS,
 L1=('C1'),
 L2=(environment),
 L3=('PMENU'),
 L4=(menuitem)
```

3. Determine the pseudo data set names based on the format described in Step 2 and the menu items indicated with an X in Step 1.

Menu Item	AP	AM	OD	QA	Data Set Names
DISPLAY	X	X	X	X	C1.DEV1.PMENU.DISPLAY
FOREGRND	X	X			C1.DEV1.PMENU.FOREGRND
BATCH	X	X			C1.DEV1.PMENU.BATCH
PACKAGE	X	X			C1.DEV1.PMENU.PACKAGE

Menu Item	AP	AM	OD	QA	Data Set Names
USER	X	X		X	C1.DEV1.PMENU.USER
ENVRMENT					C1.DEV1.PMENU.ENVRMENT
LOAD (Batch action only)					C1.DEV1.PMENU.LOAD
UNLOAD (Batch action only)					C1.DEV1.PMENU.UNLOAD
RELOAD (Batch action only)					C1.DEV1.PMENU.RELOAD
BATCH PACKAGE	X	X		X	C1.DEV1.PMENU.BATCHPKG
Menu Item	**PC**	**TS**	**AU**	**AD**	**Data Set Names**
DISPLAY	X	X		X	C1.DEV1.PMENU.DISPLAY
FOREGRND		X		X	C1.DEV1.PMENU.FOREGRND
BATCH		X		X	C1.DEV1.PMENU.BATCH
PACKAGE	X	X		X	C1.DEV1.PMENU.PACKAGE

USER	X	X		X	C1.DEV1.PMENU.USER
ENVRMENT		X		X	C1.DEV1.PMENU.ENVRMENT
LOAD (Batch action only)				X	C1.DEV1.PMENU.LOAD
UNLOAD (Batch action only)				X	C1.DEV1.PMENU.UNLOAD
RELOAD (Batch action only)				X	C1.DEV1.PMENU.RELOAD
BATCH PACKAGE	X	X		X	C1.DEV1.PMENU.BATCHPKG

Important! User access to data sets is indicated with an X in the previous example. You must have access to the data sets in order to have access to the Primary Options Panel actions.

Define Security Rules for Your Foreground Options Panel

The following three steps explain how to set up security for the Foreground Options panel at your site.

1. Plan a table that lists the Foreground Options panel menu items in the far left column and the users at your site along the top row. In the following example, an X indicates that a user has access to a menu item.

Menu Item	AP	AM	OD	QA	PC	TS	AU	AD
DISPLAY	X	X	X	X	X	X		X
ADD/UPDATE								X
RETRIEVE	X	X				X		X
GENERATE								X
MOVE								X
DELETE	X			X	X	X		X
SIGNIN	X			X	X	X		X
PRINT	X			X	X	X		X

2. Determine the format for your Foreground Options panel security rules. In order to generate a pseudo data set name in the format 'C1'.*environment.*'FORACTN'.*menuitem*, you must set up the NAMEQU entry in the Name Equates Table using this format:

```
NAMEQU FOREGROUND_OPTIONS,

 L1=('C1'),
 L2=(environment),
 L3=('FORACTN'),
 L4=(menuitem)
```

3. In the next tables, determine the pseudo data set names based on the format described in Step 2 and the menu items indicated with an X in Step 1.

Menu Item	AP	AM	OD	QA	Data Set Names
DISPLAY	X	X	X	X	C1.DEV1.FORACTN.DISPLAY
ADD/UPDATE					C1.DEV1.FORACTN.ADDUPT
RETRIEVE	X	X			C1.DEV1.FORACTN.RETRIEVE
GENERATE					C1.DEV1.FORACTN.GENERATE
MOVE					C1.DEV1.FORACTN.MOVE
DELETE					C1.DEV1.FORACTN.DELETE

Menu Item	AP	AM	OD	QA	Data Set Names
SIGNIN	X	X		X	C1.DEV1.FORACTN.SIGNIN
PRINT	X	X		X	

Menu Item	PC	TS	AU	AD	Data Set Names
DISPLAY	X	X		X	C1.DEV1.FORACTN.DISPLAY
ADD/UPDATE				X	C1.DEV1.FORACTN.ADDUPT
RETRIEVE		X		X	C1.DEV1.FORACTN.RETRIEVE
GENERATE				X	C1.DEV1.FORACTN.GENERATE
MOVE				X	C1.DEV1.FORACTN.MOVE
DELETE	X	X		X	C1.DEV1.FORACTN.DELETE
SIGNIN	X	X		X	C1.DEV1.FORACTN.SIGNIN
PRINT	X	X		X	C1.DEV1.FORACTN.PRINT

Note: User access to data sets is indicated with an X in the previous table. You must have READ access to the data sets in order to access to the Foreground Options panel actions.

Define Security Rules for Action Initiations

The following three steps explain how to set up security for action initiations at your site.

1. Design a table that lists the action initiation items in the far left column and the users at your site along the top row. In the following example, an X indicates that a user has access to the action.

Action	AP	AM	OD	QA	PC	TS	AU	AD
DISPLAY	X	X	X	X	X	X		X
ADD	X	X				X		X
UPDATE	X	X				X		X
RETRIEVE	X	X	X			X		X
GENERATE	X	X				X		X
MOVE				X	X	X		X
DELETE				X	X	X		X
SIGNIN	X	X	X			X		X

Action	AP	AM	OD	QA	PC	TS	AU	AD
SIGNOUT/ OVERRIDE		X				X		X
ARCHIVE						X		X
ALTER								X
RESTORE						X		X
PROCESSORS				X	X			X

2. Determine the format for your action initiation security rules. In order to generate a pseudo data set name in the format 'C1'.environment.system.subsystem.menuauth, you must set up the NAMEQU entry in the Name Equates Table using this format:

```
NAMEQU ACTION_INITIATION,
       L1=('C1'),
       L2=(environment),
       L3=(system),
       L4=(subsystem),
       L5=(menuauth)
```

Note: For a table showing the default authorization value for each RACROUTE request, see The Default Authorization Value (see page 86). In addition, ensure that you are not giving users access to activities that you do not want them to access.

3. In the next tables, determine the pseudo data set names based on the format described in Step 2 and the actions indicated with an X in Step 1.

Action	AP	AM	OD	QA	PC	TS	AU	AD	Data Set Names
DISPLAY	X	X	X	X	X	X		X	C1.DEV1.FINANCE.*.DISPLAY
ADD	X	X				X		X	C1.DEV1.FINANCE.*.ADD
UPDATE	X	X				X		X	C1.DEV1.FINANCE.*.UPDATE
RETRIEVE	X	X	X			X		X	C1.DEV1.FINANCE.*.RETRIEVE
GENERATE	X	X				X		X	C1.DEV1.FINANCE.*.GENERATE
MOVE				X	X	X		X	C1.DEV1.FINANCE.*.MOVE
DELETE	X	X		X	X	X		X	C1.DEV1.FINANCE.*.DELETE
SIGNIN	X	X	X			X			C1.DEV1.FINANCE.*.SIGNIN
SIGNOUT/ OVERRIDE		X		X		X		X	C1.DEV1.FINANCE.*.SIGNOVR
ARCHIVE						X		X	C1.DEV1.FINANCE.*.ARCHIVE

Action	AP	AM	OD	QA	PC	TS	AU	AD	Data Set Names
ALTER								X	C1.DEV1.FINANCE.*.ALTER
PROCESSORS						X		X	C1.DEV1.FINANCE.*.ENVRNMGR

Note: Your access to data sets is indicated with an X. You must have READ access to the data sets in order to access actions. For a description of column headings, see User Abbreviations (see page 95).

Define Security Rules for Your Package Actions Panel

The following three steps explain how to set up security for the Package Actions panel at your site.

1. Plan a table that lists the Package Actions panel menu items in the far left column and the users at your site along the top row. In the following example, an X indicates a user has access to a menu item.

Menu Item	AP	AM	OD	QA	PC	TS	AU	AD
DISPLAY	X	X	X	X	X	X		X
CREATE								X
MODIFY	X	X				X		X
CAST								X
REVIEW								X
EXECUTE	X			X	X	X		X
BACKOUT	X			X	X	X		X
COMMIT	X			X	X	X		X
UTILITY	X			X	X	X		X
SHIP								X

2. Determine the format for your Package Actions panel security rules. In order to generate a pseudo data set name in the format 'C1'.'PACKAGE'.*menuitem.pkgsubfc.pkgid*, you must set up the NAMEQU entry in the Name Equates Table using this format:

```
NAMEQU PACKAGE_ACTIONS,
       L1=('C1'),
       L2=('PACKAGE'),
       L3=(menuitem),
       L4=(pkgsubfc),
       L5=(pkgid)
```

3. In the next tables, determine the pseudo data set names based on the format described in Step 2 and the menu items indicated with an X in Step 1.

Menu Item	AP	AM	OD	QA	PC	TS	AU	AD	Data Set Names
DISPLAY	X	X	X	X	X	X		X	C1.PACKAGE.DISPLAY.APPROVER.PKG001
CREATE								X	C1.PACKAGE.CREATE.BUILD.PKG001
MODIFY	X	X				X		X	C1.PACKAGE.MODIFY.IMPORT.PKG001
CAST								X	C1.PACKAGE.CAST.CAST.PKG001
REVIEW								X	C1.PACKAGE.REVIEW.DENY.PKG001
EXECUTE					X	X		X	C1.PACKAGE.EXECUTE.EXECUTE.PKG001
BACKOUT	X	X		X	X	X		X	C1.PACKAGE.BACKOUT.BACKOUT.PKG001
COMMIT	X	X		X	X	X		X	C1.PACKAGE.COMMIT.COMMIT.PKG001
UTILITY				X	X	X		X	C1.PACKAGE.UTILITY.EXPORT.PKG001
SHIP				X	X	X		X	C1.PACKAGE.PSHIP.PSHIP.PKG001

Note: Your access to data sets is indicated with an X. You must have READ access to the data sets in order to access the Package Actions panel. For a description of column headings, see User Abbreviations (see page 95).

Define a Security Rule for Concurrent Action Processing

You can restrict who can use the concurrent action processing facility at your site.

To set up security for access to concurrent action processing at your site

1. Determine which departments or users should have access to Concurrent Action Processing.
2. Determine the format for your Concurrent Action Processing security rule. In order to generate a pseudo data set name in the format 'C1.CAP' you must set up the NAMEQU entry in the Name Equates Table using this format:

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

Site Security Package Rules

Before using ESI, you must write rules to correspond to Formats ENVIRONMENT_ACCESS through CONCURRENT_ACT_PROC. You can customize these formats to conform to your site security conventions, to alter the names generated at each control point, or to change the authority level and class.

CA Endevor SCM ESI lets you map CA Endevor SCM entities to your site security package. Different security packages such as RACF, CA ACF2 for z/OS, and CA Top Secret for z/OS have different approaches to implementing security. You need to understand the approach used by your site security package before attempting to map CA Endevor SCM entities to your site security package.

Important! All site security packages (RACF, CA ACF2 for z/OS, and CA Top Secret for z/OS) deny access if security rules are not defined.

We recommend that you consider the following guidelines when writing security rules:

- Your security rules should conform to your existing site naming conventions.
- You should define action authorities based on pseudo data set rules rather than SAF authorities.
- Simplify rule definitions by omitting unnecessary format levels.
- Avoid inadvertently creating physical data sets when working with pseudo data sets.
- You should do ONE of the following:
  - Use separate naming conventions for physical data sets and pseudo data sets.
  - Define a different security class for pseudo data sets. See Defining a Class Other Than Data Set with RACF for more information.
- If the authorization values in the SAF interface are customized you should check and possibly modify the CA Endevor SCM authorization mapping (the FUNCEQU entries).

Develop ESI Profiles

After you complete your ESI worksheet, you should use the pseudo data set names you created for the ESI worksheets to develop generic profiles. Determine whether you can combine any profiles before you submit the profiles to your site's security administrator.

For example, if you have three environments (DEV1, DEV2 and PROD) with three systems (FINANCE, PAYROLL and HUMNRCS) and you determine from the Action Initiation worksheet that QA has the authority to move elements for all three systems within the DEV1 and DEV2 environments. Assume that you originally coded the profiles as shown next:

C1.DEV1.FINANCE.*.MOVE
C1.DEV1.PAYROLL.*.MOVE
C1.DEV1.HUMNRCS.*.MOVE
C1.DEV2.FINANCE.*.MOVE
C1.DEV2.PAYROLL.*.MOVE
C1.DEV2.HUMNRCS.*.MOVE

Notice that in every case only the second and third qualifiers are different. The second qualifier must be specified because it defines the environment. The PROD environment is not included for QA access. The third qualifier, however, includes all systems so that you can make it generic with a wildcard character as shown in the following table:

These three profiles	Becomes one of these two profiles
C1.DEV1.FINANCE.*.MOVE	C1.DEV1.*.*.MOVE
C1.DEV1.PAYROLL.*.MOVE	or
C1.DEV1.HUMNRCS.*.MOVE	C1.DEV.-.-.MOVE
C1.DEV2.FINANCE.*.MOVE	C1.DEV2.*.*.MOVE
C1.DEV2.PAYROLL.*.MOVE	or
C1.DEV2.HUMNRCS.*.MOVE	C1.DEV2.-.-.MOVE

Identify User IDs that Require Access to ESI Rules and Profiles

After you complete your ESI worksheet, you need to supply the TSO IDs that are required to access the ESI rules/profiles you created. Whenever possible you should identify logical groups of IDs as shown in the following example:

```
all QA1* Ids
```

Create NAMEQU Entries for the Name Equates Table

Use the ENVIRONMENT_ACCESS through CONCURRENT_ACT_PROC (extension ACTION_INITIATION is optional) layouts you developed for the ESI worksheets to create NAMEQU entries for the Name Equates Table. Edit member BC1TNEQU of your iprfx.iqual.CSIQSRC installation library and modify the entries to reflect your choices for each of the formats.

The following format	Is derived from
ENVIRONMENT_ACCESS	The Environment Worksheet (Part 3)
PRIMARY_OPTIONS	The Primary Options Worksheet (Part 2)
FOREGROUND_OPTIONS	The Foreground Options Worksheet (Part 2)
ACTION_INITIATION	The Action Initiation Worksheet (Part2)
ACTION_INITIATION (extension is optional)	The Action Initiation Worksheet (Part2)
PACKAGE_ACTIONS	The Package Actions Worksheet

How to Assemble and Link the Name Equates Table

After you determine the ESIDFLTS, FUNCEQU and NAMEQU entries for your Name Equates Table you must complete the following four tasks:

- Define the rules to your site security package.
- Modify the BC1TNEQU member in your iprfx.iqual.CSIQSRC.
- Assemble and link the modified table, using an SMP/E USERMOD. Alternatively, edit the sample JCL BC1JTABL, located in iprfx.iqual.CSIQJCL, and use it to assemble and link source module BC1TNEQU outside of SMP/E.
- Refresh the LINKLIST if the library is in the LINKLIST.

Define Rules for Your Site Security Package

You should ensure that security rules are defined to your site security package. You can define security rules prior to enabling ESI.

Important! If you enable ESI without defining security rules, all access is denied.

Modify the BC1JNEQU Member

Edit member BC1JNEQU in your iprfx.iqual.CSIQJCL and modify the Name Equates Table entries to reflect the parameters you selected for your site. Ensure that the table name ESIDFLTS entry label matches the load module name created in the linkage step with the SYSLMOD DD statement.

Assemble and Linking the Modified Table

Perform an LLA REFRESH if your authorized library is in the LINKLIST. Note that the name on the ESIDFLTS entry label and the member name on the SYSLMOD DD statement match.

At run time, ESI dynamically loads the newly created Name Equates Table from an authorized library (either in LINKLIST or STEPLIB) to determine the RACROUTE request values to use.

Refresh the LINKLIST

Perform an LLA REFRESH if your authorized library is in the LINKLIST. The JCL that assembles and links the Name Equates Table and source code can be found in the member BC1NEQU. The table name has been modified to NEWTNEQU. Note that the name on the ESIDFLTS entry label and the member name on the SYSLMOD DD statement match.

At run time, ESI dynamically loads the newly created Name Equates Table from an authorized library (either in LINKLIST or STEPLIB) to determine the RACROUTE request values to use.

Activate ESI Using the C1DEFLTS Table

The C1DEFLTS table establishes the site, environment, and stage definitions for your installation. This table is an assembler macro that defines parameters specific to the site as a whole, and to each environment and stage at the site. Use the following fields in the C1DEFLTS table to activate ESI:

ACCSTBL

Contains the name of the Name Equates table. The default value for this field is BC1TNEQU.

ESSI

Validates your purchase of ESI and indicates if you want to use ESI security or native mode security. The default value for the ESSI parameter is N. This indicates that you did not purchase ESI and that you are using native mode security. To indicate that you want to use ESI security you must enter a 'Y' or 'N' in the ESSI field in the C1DEFLTS Table.

PKGSEC

The PKGSEC keyword has three options:

APPROVER

Indicates that standard approver security should be used (pre-Release 3.8).

ESI

Indicates that the ESI interface is used, with the exception of review processing which still requires the approver group (internal or external).

MIGRATE

Allows both methods to be performed, with the Approver method having priority over the ESI method. Use this method if you're migrating to the ESI method.

You can have external approver groups and package security through ESI.

- APPROVER - Restricts package action through approver groups whether it is internal or external to CA Endevor SCM.
- ESI - Controls package actions with external security packages CA ACF2 for z/OS, CA Top Secret for z/OS, and RACF via the ESI interface.
- MIGRATE - Package actions are controlled by approver groups and/or ESI security.

These options are different in that:

- PKGSEC=APPROVER allows you to restrict package actions through approver groups.
- If PKGSEC=ESI in the C1DEFLTS table, all package actions, including APPROVAL invoke ESI. To approve a package, the user must be a member of the approver group whether it is an internal or external group. If you are a member of the approver group, an ESI call is made to check if you are authorized to perform the REVIEW action. You can perform the action if you are authorized, otherwise the action is denied.

  Note: If No approval groups are related to an inventory location, the package is automatically approved. QUORUM must be set to one or more.
- PKGSEC=MIGRATE - Once a package is created, cast and approved the rules for ESI are invoked. The first security call after the package is approved, is a call to the approver group. If you are a member of the approver group, the package action is allowed. If you are not a member of the approver group, an ESI call is made to see if you are authorized to perform the action. If you are authorized, the action is granted. If you are not authorized, the action is denied.

Internal and/or external approver groups are required for approval processing. If no approval groups are related to an inventory group, the package is automatically approved. QUORUM must be set to one or more.

Note: For more information about setting up external approver groups under CA ACF2 for z/OS, RACF, and CA Top Secret for z/OS, see the *Packages Guide*.

Modify the Name Equates Table Name

You can change the name of the Name Equates Table by changing the value in the ACCSTBL field. You must have also assembled and linked the Name Equates Table with an identical CSECT name and load module name. For example, if ACCSTBL=NEWTNEQU is coded in the C1DEFLTS Table, the load module named NEWTNEQU is assumed to be the name for the Name Equates Table and the CSECT within the load module must have the same name. The Name Equates Table contains a field that is set at assembly time with the name of the CSECT. This prevents a user from copying and renaming an invalid Name Equates table in order to gain unauthorized access.

Note: Do not enable ESSI or enter a table name (ESSI and ACCSTBL) until you are ready to test ESI.

In the following C1DEFLTS Table, note the ESSI PKGSEC and ACCSTBL fields. The ACCSTBL field is set to a Name Equates Table name of NEWTNEQU. PKGSEC is set to a value of ESI, and the ESSI enabled flag is set to a value of 'Y'.

```
C1DEFLTS TYPE=MAIN,
        ACCSTBL=NEWTNEQU,   ACCESS SECURITY TABLE            *
        APRVFLG=N,          APPROVAL PROCESSING (Y/N)        *
        ESSI=Y,             ESI ENABLED                      *
        PKGCSEC=Y,          PACKAGE CAST SECURITY            *
        PKGISEC=Y,          PACKAGE INSPECT SECURITY         *
        PKGSEC=ESI,         USE EXTERNAL SECURITY            *
        RACFUID=,           ALTERNATE ID USERID              *
```

The first portion of the C1DEFLTS Table (TYPE=MAIN) should be set up only once.

Note: For more information about these fields and the C1DEFLTS table, see the *Administration Guide*.

Test ESI Security and Monitor Warnings

Once you have successfully installed and enabled CA Endevor SCM ESI, you need to verify that ESI has been activated to monitor the security violations issued by CA Endevor SCM.

Note: For more information about testing ESI, see The ESI Warning Mode (see page 109).

Verify ESI Activation

To ensure CA Endevor SCM ESI is successfully enabled, display your site information. If a **Y** appears in the ESI field, ESI is enabled. The Access Table field contains the name of the Name Equates Table and the PKGSEC field is set to **Y**.

Monitor Security Violations

You can monitor security violations using ESI warning mode or the CONRPT40 and CONRPT41 reports. Warning mode violations are recorded in the ESI warning report. For more information on warning mode, see Using ESI Warning Mode.

The CONRPT40 and CONRPT41 reports record attempts by users to perform unauthorized actions.

Note: For more information about using CONRPT40 and CONRPT41, see the *Reports Guide.*

The ESI Warning Mode

ESI Warning Mode allows you to test your security implementation without denying users access to CA Endevor SCM objects. If access rules are not coded, CA Endevor SCM ordinarily denies access. When using Warning Mode, access to resources is allowed even if your site security package (RACF, CA ACF2 for z/OS, or CA Top Secret for z/OS) indicates that access should be denied. You should use ESI Warning Mode *before* writing security rules or as an initial test for your security rules.

When your security system identifies a security exception and ESI Warning Mode is enabled, a System Management Facility (SMF) instance records the event. You can then use the ESI Exception Warning report to format and print the ESI warning SMF records in a convenient and easy-to-read report.

SMF records are always written to the SMF data sets (SYS1.MANx). In addition, you can send records to a sequential data set by allocating a data set to the DDname EN$SMESI. This automatically writes the SMF records to the SMF data set as well the sequential data set.

The ESI Exception Warning report summarizes the SMF records written to the SMF and the sequential data sets. The report data is summarized by entity name, entity format type, entity class, user ID, event date and time. The original return codes and reason codes are listed as well as a summary report including each entity name and all the exceptions associated with the entity. The Exception Warning report also includes information about each Name Equates Table encountered.

The following JCL example shows how to execute an Exception Warning report:

```
//ESIWREPT EXEC PRM=NDVRC1,PARM='ENRASW00',REGION=4096K
//STEPLIB DD  DISP=SHR,DSN=iprfx.iqual.loadlib
//EN$SMESI DD  DISP=SHR,DSN=SMF.data.set
//EN$SMRPT DD  SYSOUT=*,DCB=(LRECL=133,BLKSIZE=6118,RECFM=FBA)
```

In this example, the program NDVRC1 invokes the exception report ENRASW00. EN$SMESI identifies the input SMF records and EN$SMRPT identifies the output data set.

The ESI Defaults (ESIDFLTS) Macro

The ESI Defaults macro (ESIDFLTS) allows you to specify how you manage ESI diagnostics. You can use the ESIDFLTS macro to write a diagnostic trace to improve performance and enable warning mode.

Enable ESI Warning Mode

To enable ESI warning mode, you must specify WARN=YES in the Name Equates Table. The keyword is specified on either the ESIDFLTS or NAMEQU entries. The ESIDFLTS value affects the entire table. The NAMEQU value only affects the entry for which it is specified and overrides the ESIDFLTS value.

The ESI Trace Facility

The ESI Trace Facility helps you to determine if security is functioning as desired at your site. When the Trace Facility is activated, every security call to ESI writes a trace record. The Trace DDname determines where the data is sent. You can use the Trace Facility to perform the following tasks:

- Ensure that the format of the entity name is correct.
- Review the results of a SAF request (a return code or a reason code).

Ensure the Correct Format of the Pseudo Data Set Name

You can examine a trace record to determine if the format of the entity name is correct. The following sample trace record shows a highlighted pseudo data set name (Entity=C1.ENVIRON.PROD).

```
ENCS001I:  Using BC1TNEQU table entitled 'CA ENDEVOR NAMEQU Table 4.0'
ENCS001I:  The table was assembled on 07/18/00 at 12.09
ENCS001I:  ESI defaults:
ENCS001I:  ESIDFLTS DESC=(6),                                                +
ENCS001I:     ROUTCDE=(11),                                                  +
ENCS001I:     WARN=YES,                                                      +
ENCS001I:     HEADER=YES,                                                    +
ENCS001I:     LATSIZE=10,                                                    +
ENCS001I:     TITLE='CA ENDEVOR NAMEQU Table 4.0'
ENCS001I:  Function authorization equates:
ENCS001I:  FUNCEQU SAFAUTH=NONE,                                             +

ENCS001I:     C1ACTNS=(PRINT,SIGNIN)
ENCS001I:  FUNCEQU SAFAUTH=CONTROL,                                          +
ENCS001I:     C1ACTNS=(ARCHIVE,DELETE,DISPLAY,MOVE,RETRIEVE,SIGNOVR),        +
ENCS001I:  FUNCEQU SAFAUTH=UPDATE,                                           +
ENCS001I:     C1ACTNS=(ADD,GENERATE,UPDATE)
ENCS001I:  FUNCEQU SAFAUTH=ALTER,                                            +
ENCS001I:     C1ACTNS=(ENVRNMGR)
ENCS001I:  Format definitions:
ENCS001I:  NAMEQU ENVIRONMENT_ACCESS,                                        +
ENCS001I:     CLASS='$ENDEVOR',                                              +
ENCS001I:     WARN=NO,                                                       +
ENCS001I:     LOG=NONE,                                                      +
ENCS001I:     L1=('CA ENDEVOR'),                                             +
ENCS001I:     L2=('CLASS=$ENDEVOR')                                          +
ENCS001I:     L3=('ENVIRONMENT_ACCESS_TEST'),                                +
ENCS001I:     L4=('ENVIRONMENT=',ENVIRONMENT)
ENCS101I Format=0001 Pass=0000 Auth=READ ACEE=00000000
ENCS101I Class=$ENDEVOR Log=NONE
ENCS101I Scale=0....+....1....+....2....+....3....+....4....+....5....+....6
ENCS101I Entity=CA ENDEVOR.CLASS=$ENDEVOR.ENVIRONMENT_ACCESS_TEST.ENVIRONMENT=D

        ....+....7....+....8....+....9....+....0....+....1....+....2
        EV
ENCS101I User DA2DM47 access is denied  from SAF
ENCS101I RACROUTE RC=0008 RACHECK RC=0008 REASON=0000
ENCS101I Format=0001 Pass=0000 Auth=READ ACEE=00000000
ENCS101I Class=$ENDEVOR Log=NONE
ENCS101I Scale=0....+....1....+....2....+....3....+....4....+....5....+....6
ENCS101I Entity=CA ENDEVOR.CLASS=$ENDEVOR.ENVIRONMENT_ACCESS_TEST.ENVIRONMENT=B
        ....+....7....+....8....+....9....+....0....+....1....+....2
        ST
ENCS101I User DA2DM47 access is allowed from SAF
```

Review RACROUTE Request Return Codes

You can examine a trace record to review RACROUTE request return codes. The previous sample trace record includes highlighted RACROUTE return codes (RACROUTE(0000) RACHECK(0000) REASON(0000)).

The Trace Record Format

The Trace Facility trace records have the following format:

```
FORMAT=n
PASS=n
AUTH=n
ACEE=n
CLASS=DATASET
ENTITY=(Ln...)
USER=userid
ACCESS=(ALLOWED/DISALLOWED)
FROM=(SAF/LAT)
in WARN mode
RACROUTE RC=n
RACHECK RC=n
REASON RC=n
```

Format=n

Specifies the NAMEQU format, where:

- FORMAT1 is ENVIRONMENT_ACCESS
- FORMAT2 is PRIMARY_OPTIONS
- FORMAT3 is FOREGROUND_OPTIONS
- FORMAT4 and FORMAT5 are ACTION_INITIATION
- FORMAT6 is PACKAGE_ACTIONS
- FORMAT7 is CONCURRENT_ACT_PROC

PASS=n

Always 0 or 1.

AUTH=auth1

Specifies the requested authorization value as defined by the FUNCEQU macro: 'NONE' 'READ' 'UPDT' (UPDATE) 'CNTL' (CONTROL) 'ALTR' (ALTER)

ACEE=n

For TSO environments this value is usually 0. For ROSCOE environments this value is the virtual storage address of the ACEE passed to the SAF interface.

CLASS=class name

Specifies the resource name security class. The default is class name.

ENTITY=entity name

Specifies the pseudo data set name passed to SAF.

USER=userid

Specifies a user ID for verification.

ACCESS=(ALLOWED/DISALLOWED)

Specifies whether access to the entity is allowed or disallowed.

FROM=(SAF|LAT)

Specifies either SAF or LAT.

in WARN mode

Displayed only when running in warning mode. Specifies that a user is allowed access because warning mode is turned on.

RACROUTE RC=n

Specifies the return code from the RACROUTE request: 0 = Permit request, non-zero = Fail request

RACHECK RC=n

Specifies the return code from the RACHECK request (internal to RACROUTE).

REASON=n

Specifies the reason code from the RACHECK.

Note: For CA ACF2 for z/OS, and CA Top Secret for z/OS users, the AUTH value is alternately mapped to the appropriate value.

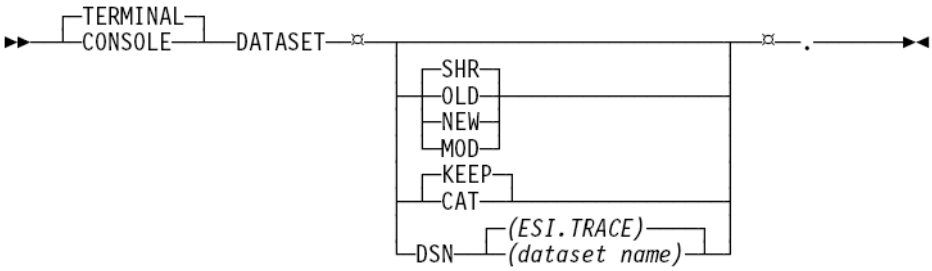
Use the Trace Facility

You must enable CA Endevor SCM ESI and have access to the CA Endevor SCM CLIST Library in order to use the ESITRACE command or DD statement. If you do not have access to the CLIST Library, you can use the TSO ALLOCATE command.

Note: For more information about using the TSO ALLOCATE command, see TSO ALLOCATE Command Tasks (see page 116).

ESITRACE Command in Foreground Mode

The ESITRACE command makes the Trace Facility easy-to-use. The ESITRACE syntax is shown next.



Use the following ESITRACE parameters to activate and deactivate the ESI Trace Facility in foreground mode.

Start

Allocates the trace data set which activates the Trace Facility. START is the default.

Stop

Deallocates the trace data set which deactivates the Trace Facility.

TERMINAL

Directs the trace output to your terminal. TERMINAL is the default.

CONSOLE

Directs the trace output to the operator's console.

DATASET

Allocates the trace output to the data set identified by the DSN parameter:

SHR

Allocates the trace data set with share status. SHR is the default.

OLD

Allocates the trace data set exclusively.

NEW

Allocates the trace data set with the name specified in the DSN parameter.

MOD

Writes new trace records at the end of the trace data set (EN$TRESI).

KEEP

Keeps the trace data set when deallocated. KEEP is the default.

CAT

Catalogs the trace data set (EN$TRESI) when deallocated.

DSN (data set name)

Identifies the name of the trace data set and is ignored for CONSOLE or TERMINAL allocations. The DSN follows the rules for TSO data set names.

ESI.TRACE

The default name of the trace data set.

dataset name

The user specified trace data set name.

How to Run ESI Trace

You can write trace records to a terminal, the operator's console or to data sets as shown in the following three examples:

1. Use the following CLIST parameter to start the Trace Facility and send all trace records to your terminal by default:

   ```
   %ESITRACE START
   ```

2. Use the following CLIST parameter to start the Trace Facility and write trace records to the operator's console:

   ```
   %ESITRACE CONSOLE START
   ```

3. Use the following CLIST parameter to start the Trace Facility and write all trace records to the data set named userid.ESI.TRACE:

   ```
   %ESITRACE DATASET DSN (ESI.TRACE) NEW CAT
   ```

Note: You must preface the ESITRACE command with 'TSO' if you enter commands from an ISPF screen.

Deactivate the Trace Facility

You should deactivate the Trace Facility after you have determined that ESI is properly configured. Use the following CLIST parameter to deactivate the Trace Facility.

```
%ESITRACE STOP
```

TSO ALLOCATE Command Tasks

The following table shows how you can use the TSO ALLOCATE and FREE commands to enable and disable the ESITRACE facility:

Task	TSO ALLOCATE/FREE Command
Start the Trace Facility	ALLOC DD (EN$TRESI) DA(MY.ESI.DATASET) NEW CAT SPACE (11) CYLINDERS LRECL(133) BLKSIZE(6118) RECFM(FBA) UNIT(SYSDA) VOL(TSO001)
Allocate a data set to your terminal	ALLOC DD (EN$TRESI) DA(*) SHR
Allocate an existing data set and append records	ALLOC DD (EN$TRESI) DA(MY.ESI.TRACE.DATASET) MOD
Write trace data to the operator's console	ALLOC DD (EN$TRESI) DUMMY SHR
Stop the Trace Facility	FREE DD (EN$TRESI)

If you allocate the ESI trace using the TSO ALLOCATE command or ISPF panels, you must use the following the DCB parameters:

- RECFM (FBA)
- LRECL (133)
- BLKSIZE (multiples of 133)

Note: For more information about the TSO ALLOCATE command, see your TSO documentation.

Activate ESI Trace in Batch Mode

You can activate the ESI Trace facility for batch processing by including the following DD statement in your execution JCL:

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

You can write trace information to a data set by putting the following DD statement in your execution JCL:

```
//EN$TRESI DD DSN=data.set.name,DISP=SHR
```

Where DCB attributes match the ones noted in the previous code.

Appendix A: Security Worksheets

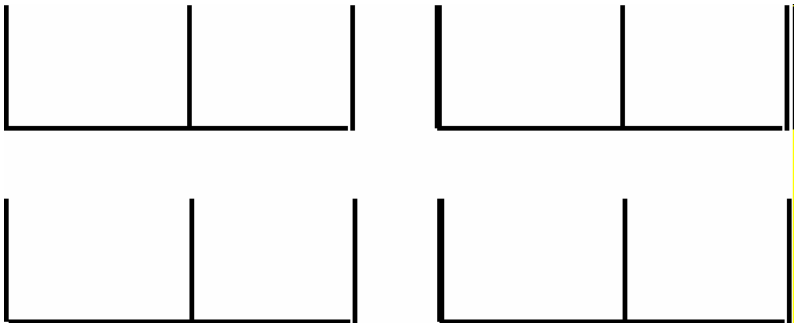
This section contains the following topics:

The Environment Security Control Worksheet

Use the different parts of this worksheet to define security rules for your site.

Part 1

Label and map your environment in the following table.



Part 2

Enter environment names and indicate access with an X in the appropriate boxes.

Env\User
ENV1:
ENV2:
ENV3:
ENV4:

Part 3

In the spaces provided, determine the format for your Environment Rule.

(L1)_____ (L2)_____ (L3)_____

Part 4

In the following spaces, determine pseudo dataset names based on the environment rule in Part 3 and the environments entered in Part 2. For example:

```
C1.ENVIRON.PROD
```

Note: User must have READ access to this dataset in order to have environment access.

The Primary Options Security Control Worksheet

Use the different parts of this worksheet to define security rules for your site.

Part 1

In the spaces provided, provide a name for the elements.

Environment: _____

System: _____

Subsystem: _____

Part 2

In the following table, indicate the user's access to the Primary Options Menu item by marking an X in the appropriate box.

Primary Menu Item\User
DISPLAY
FOREGRND
BATCH
PACKAGE
USER
ENVRMENT
LOAD
UNLOAD
RELOAD
BATCHPKG

Part 3

In the spaces provided, determine the format for your Primary Options Rule.

(L1)_____ (L2)_____ (L3)_____ (L4)_____

Part 4

In the spaces provided, determine pseudo dataset names; for example:

```
C1.PROD.PMENU.DISPLAY
```

Note: You must have READ access to the dataset in order for this option to appear on the Primary Options Menu.

The Foreground Options Security Control Worksheet

Use the different parts of this worksheet to define security rules for your site.

Part 1

In the spaces provided, provide a name for the elements.

Environment: _____

System: _____

Subsystem: _____

Part 2

In the following table, indicate the user's access to the Foreground Options Menu item by marking an X in the appropriate box.

Frgd Menu Item\User
DISPLAY
ADDUPDT
RETRIEVE
GENERATE
MOVE
DELETE
SIGNIN
PRINT

Part 3

In the spaces provided, determine the format for your Foreground Options Rule.

(L1)_____ (L2)_____ (L3)_____ (L4)_____

Part 4

In the spaces provided, determine pseudo dataset names; for example:

```
C1.PROD.FORACTN.GENERATE
```

Note: You must have READ access to the dataset in order for this option to appear on the Foreground Options Menu.

The Action Initiation Security Control Worksheet

Use the different parts of this worksheet to define security rules for your site.

Part 1

In the spaces provided, provide a name for the elements.

Environment: _____

System: _____

Subsystem: _____

Part 2

In the following table, indicate the user's authority to perform an action for the stage indicated by marking an X in the appropriate box.

Action Menu Item\User
DISPLAY
ADD
RETRIEVE
GENERATE
MOVE
DELETE
SIGNOVR
ARCHIVE
ALTER
ENVRNMGR

Part 3

In the spaces provided, determine the format of your Action Initiation Rule.

(L1)_____ (L2)_____ (L3)_____ (L4)_____

Part 4

In the spaces provided, determine pseudo dataset names; for example:

```
C1.PROD.FINANCE*.MOVE
```

Note: You must have READ access to the dataset in order for this option to appear on the Primary Options Menu.

The Package Actions Security Control Worksheet

Use the different parts of this worksheet to define security rules for your site.

Part 1

In the following table, indicate the user's authority to perform an action for the stage indicated by marking an X in the appropriate box.

Package Menu Item Subfunction\User
DISPLAY
Blank
Backout
Approvers
SCL
Reports
CREATE
Build
Import
Edit
Copy
LIST*
MODIFY
Build
Import

Package Menu Item **Subfunction\User**
Edit
Copy
CAST
Cast
SCL
REVIEW
Blank
Deny
Approve
List
SHIP
Xmit
BACKOUT
Backout
Display
Backin
COMMIT
Commit
DYNAMIC
Add
Update
Delete
UTILITY
Display
Blank
Backout
Approvers
SCL
Reports

Package Menu Item Subfunction\User
Export
Reset
Delete

Note: ***LIST** does not display as a menu item, but it is used by CA Endevor SCM to build available package lists. There is no security for the Notes subfunction.

Part 2

In the spaces provided, determine the format of your Package Actions Rule.

(L1)_____ (L2)_____ (L3)_____ (L4)_____ (L5)_____

Part 4

In the spaces provided, determine pseudo dataset names; for example:

C1.PACKAGE.CREATE.*.PGK*

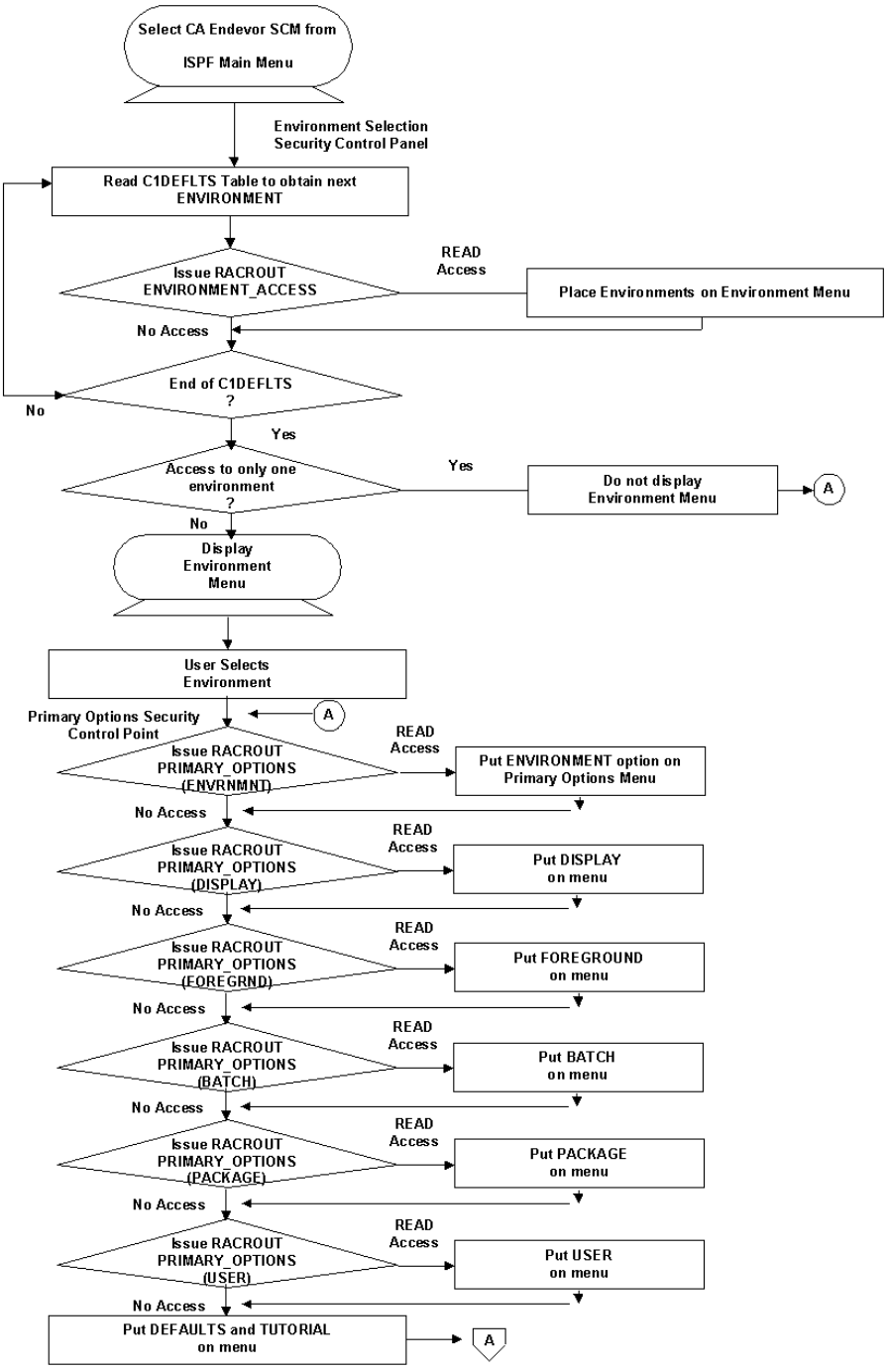
Note: You must have READ access to the dataset in order for this option to appear on the Primary Options Menu.

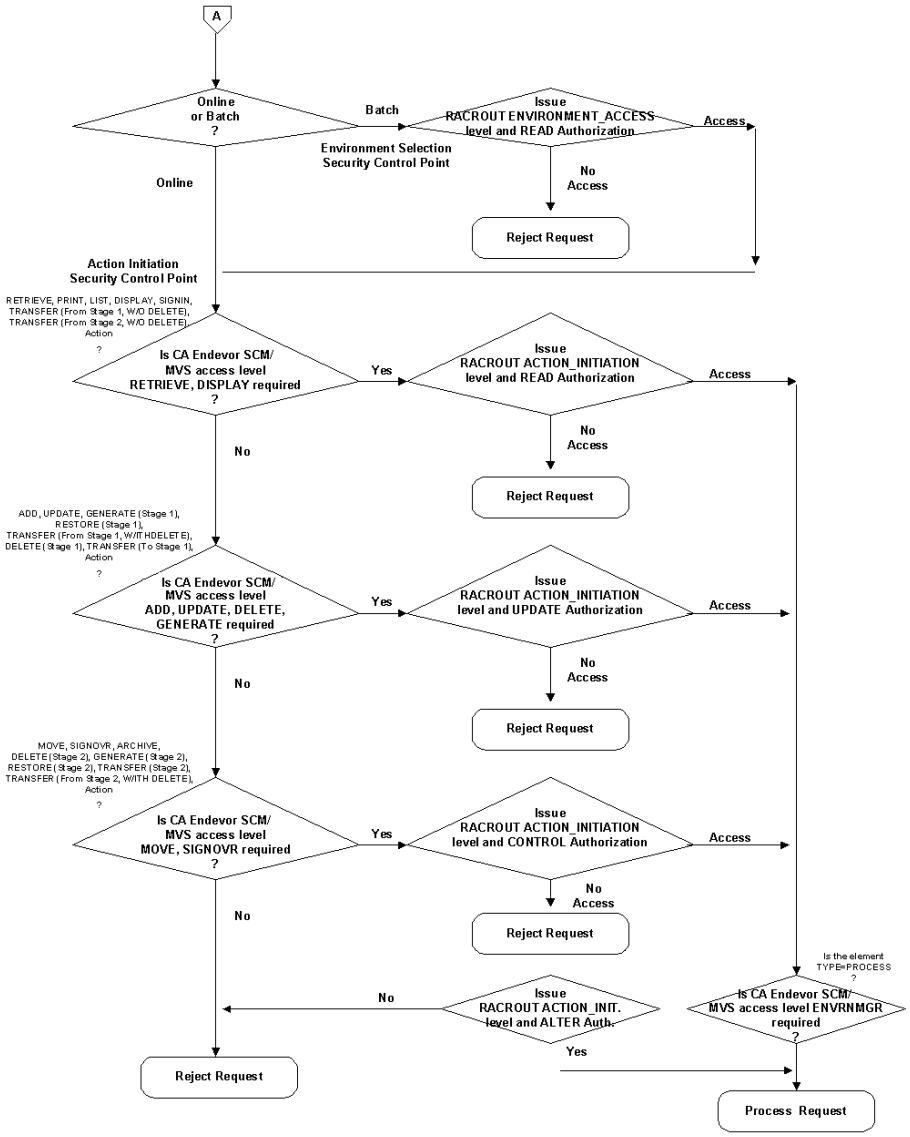
Appendix B: ESI Logic Flow

This section contains the following topics:

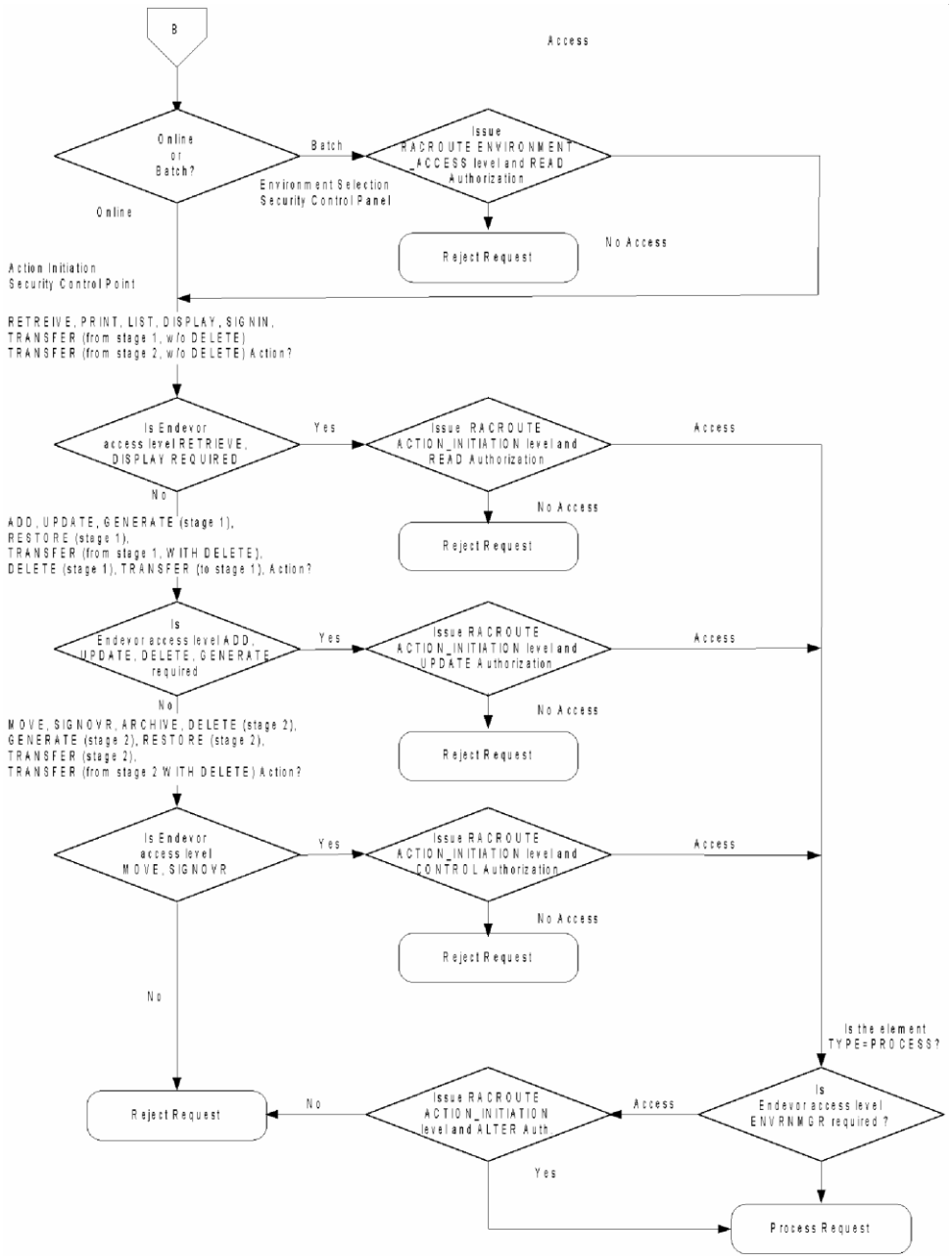
ESI Logic Flow Diagram

The following diagram shows the logic flow of CA Endevor SCM security decisions and FORMAT name generation. Please note that this flow chart illustrates default settings. You can alter these settings with MACRO definitions.




A
Online or Batch ?
Batch
Issue RACROUT ENVIRONMENT_ACCESS level and READ Authorization
Access
Environment Selection Security Control Point
No Access
Online
Reject Request
Action Initiation Security Control Point
RETRIEVE, PRINT, LIST, DISPLAY, SIGNIN, TRANSFER (From Stage 1, W/O DELETE), TRANSFER (From Stage 2, W/O DELETE), Action ?
Is CA Endevor SCM/ MVS access level RETRIEVE, DISPLAY required ?
Yes
Issue RACROUT ACTION_INITIATION level and READ Authorization
Access
No Access
No
Reject Request
ADD, UPDATE, GENERATE (Stage 1), RESTORE (Stage 1), TRANSFER (From Stage 1, W/IT HDELETE), DELETE (Stage 1), TRANSFER (To Stage 1), Action ?
Is CA Endevor SCM/ MVS access level ADD, UPDATE, DELETE, GENERATE required ?
Yes
Issue RACROUT ACTION_INITIATION level and UPDATE Authorization
Access
No Access
No
Reject Request
MOVE, SIGNOVR, ARCHIVE, DELETE (Stage 2), GENERATE (Stage 2), RESTORE (Stage 2), TRANSFER (Stage 2), TRANSFER (From Stage 2, W/ITH DELETE), Action ?
Is CA Endevor SCM/ MVS access level MOVE, SIGNOVR required ?
Yes
Issue RACROUT ACTION_INITIATION level and CONTROL Authorization
Access
No Access
No
Reject Request
Is the element TYPE=PROCESS ?
Is CA Endevor SCM/ MVS access level ENVRNMGR required ?
No
Issue RACROUT ACTION_INIT. level and ALTER Auth.
Yes
Reject Request
Process Request

ESI logic flow diagram 3 of 3



Index

E

F

I

K

L

N

P

R

S

T

U

W