CA Endevor® Software Change Manager

InfoMan Interface Administration Guide

Version 17.0.00

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 InfoMan Interface (CA Endevor InfoMan Interface)

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 15.0

- Available $REQPDS Fields (see page 73)—Updated to add the REQORGH field for RGENHIST (the Restore action option Retain Generate History).

Contents

Chapter 3: Setting Up the CA Endevor SCM INFO Table 29

Chapter 1: Introduction

This section contains the following topics:

What You Need to Know

Implementing the CA Endevor InfoMan Interface to IBM Tivoli Information Management (InfoMan) requires that you make decisions about which CA Endevor SCM information you want to track in InfoMan, and where you want to store that information in InfoMan. Therefore, to implement the interface most effectively, you should have the following resources available:

- People familiar with how CA Endevor SCM is used at your site, and in particular with CA Endevor SCM packages and exits.
- People familiar with InfoMan policies and operations at your site.

You can find information in the following resources:

- *Exits Guide*
- *Packages Guide*
- *IBM Tivoli Information Management Application Program Interface Guide*

CA Endevor InfoMan Interface

The CA Endevor InfoMan Interface offers a flexible mechanism for implementing InfoMan policies as they relate to CA Endevor SCM, for controlling CA Endevor SCM actions based on InfoMan information, and for logging CA Endevor SCM information in InfoMan.

The interface works with both native InfoMan and the InfoMan Interactive Facility (IIF).

In addition to managing the initialization and termination of InfoMan sessions under CA Endevor SCM, the interface implements package-level and action-level interaction between CA Endevor SCM and InfoMan. This allows you to:

- Use the InfoMan approval process for moving packages into production, rather than the CA Endevor SCM process.
- Use the InfoMan record as an auditing tool.
- Display package information from InfoMan during a CA Endevor SCM session.
- Produce lists of available CCIDs when requesting stand-alone actions in foreground or batch SCL generation.

The interface does this by communicating directly from CA Endevor SCM exit points with the InfoMan API. The communication is accomplished through a user-defined rule set, called the E/INFO table, created when the interface is implemented. You create the rule set with a batch utility provided with the interface.

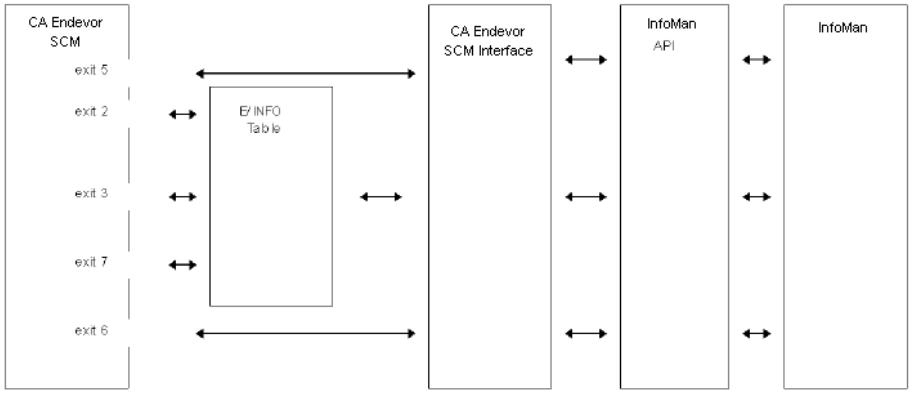
Before the existence of the InfoMan API, Terminal Simulation Programs (TSPs) were the only way to get to InfoMan. The batch utility offers another, flexible way to build a custom rule set for implementing InfoMan policies in CA Endevor SCM, and getting data requirements and processing logic from CA Endevor SCM to InfoMan.

The batch utility allows this rule set to be independent of InfoMan invocation. Moreover, the utility does not require extensive experience with either TSPs or assembler programming.

CA Endevor InfoMan Interface Operation

CA Endevor SCM provides a full set of exit points (before-exit and after-exit) for actions and packages and uses control blocks to provide information to user programs written for these exit points. InfoMan provides an API that uses data structures called PIDT tables to provide logical views of InfoMan database records. The interface operates between the CA Endevor SCM exit facility and the InfoMan API.

The diagram below summarizes interface operation.



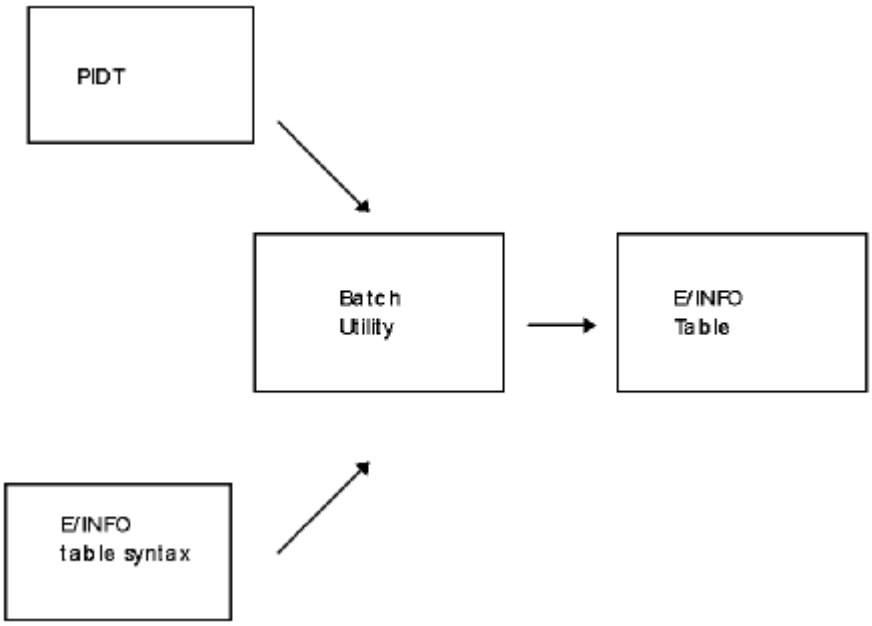
As previously illustrated, the interface communicates directly with the InfoMan API at exit 5 (initialization) and exit 6 (termination). For exits 2 (before-action), 3 (after-action), and 7 (package functions) the E/INFO table, the load module built using the batch utility, provides a bridge between the CA Endevor SCM exit facility and the InfoMan API. The rule set contained in the E/INFO table determines how the interface acts.

The CA Endevor InfoMan Interface Batch Utility

You use the batch utility (program C1BMEI00) to build an assembled rule set to drive the package and action levels of the interface. The utility provides a language that allows you to define this rule set by:

- Identifying InfoMan actions to be performed (Inquiry, Retrieve, Create, Update), and the PIDT table needed for each function.
- Specifying the exit points where the actions are to be performed.
- Establishing criteria on the CA Endevor SCM or the InfoMan side for performing an action.
- Specifying the CA Endevor SCM information to be recorded in InfoMan, and the InfoMan fields where the information is to be recorded.

The diagram below illustrates this process.



As previously illustrated, the utility assembles the definition syntax into a load module, referred to in this manual as the E/INFO table. The E/INFO table is loaded at CA Endevor SCM initialization, and is independent of InfoMan invocation. The rule set it contains determines how the interface uses the InfoMan API.

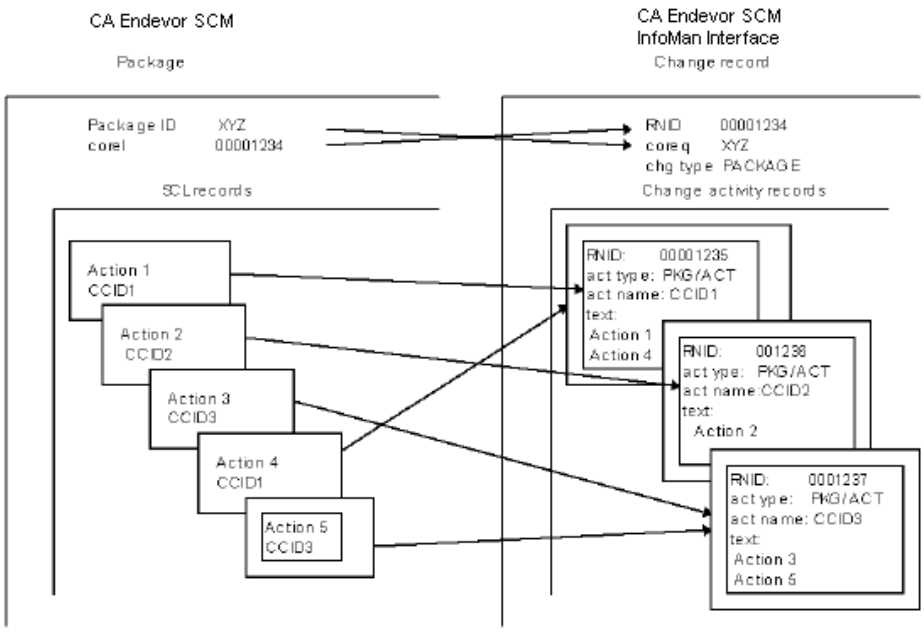
When the table is invoked, CA Endevor SCM builds a request for the InfoMan API using the information in the table. The InfoMan API executes the requested functions against the InfoMan database.

The CA Endevor InfoMan Interface Package Level

The package level of the interface allows the capture of both package life cycle information and information about the individual actions in the package. The interface establishes the following relationships:

- The CA Endevor SCM package is cross-referenced to an InfoMan change record, with the package ID stored in the COREQUISITE field of the InfoMan record, and the InfoMan RNID stored in the CORRELATION field of the CA Endevor SCM package record.
- For each unique CCID in the package action SCL records, the interface maintains a change activity record in InfoMan, with the CA Endevor SCM CCID stored in the ACTION NAME field in the InfoMan record. Information about each action associated with the CCID can be stored as freeform text in that activity record. The freeform text field is useful for this purpose because new information is always appended to existing information in the field.

The diagram below summarizes the package level of the interface.



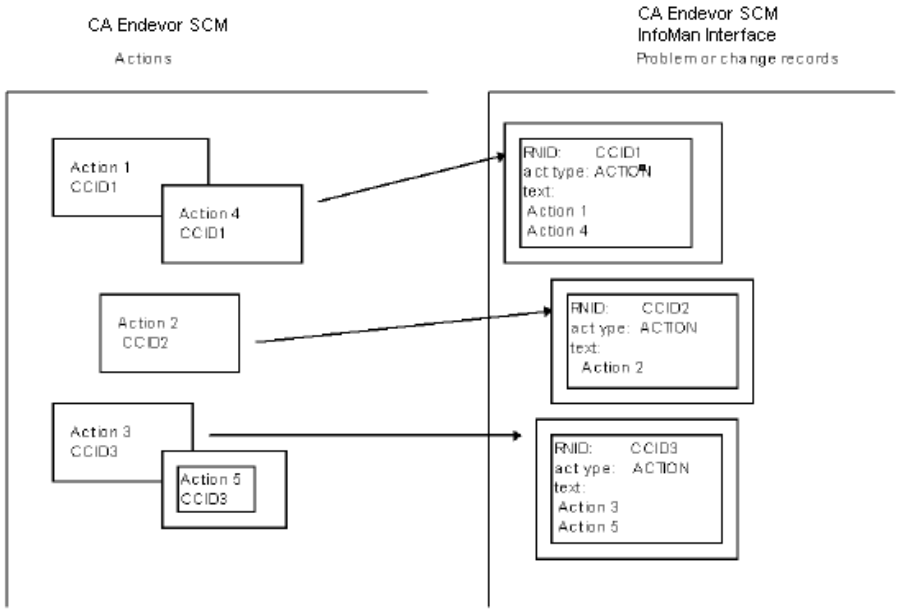
As previously illustrated, the package level of the InfoMan Interface allows sites that use InfoMan as the mechanism for approving packages for production turnover to use the InfoMan approval process instead of the approval process for CA Endevor SCM packages.

The interface also provides an improved audit trail for packages, and better reporting on packages than is currently available from CA Endevor SCM.

Note: You can view package information stored in InfoMan using reports. For more information, see Using CA Endevor InfoMan Interface Reports (see page 133). In addition, in foreground you can view package correlation data from the Package Display panel.

The CA Endevor InfoMan Interface Action Level

Sites that do not use packages, or prefer to use InfoMan Problem records, can utilize the action level of the interface. For standalone CA Endevor SCM actions, the interface creates either a Problem or a Change record in InfoMan for each unique CCID. The diagram below shows this relationship.



As previously illustrated, you can either utilize the action level of the interface, or you can create either a Problem or a Change record in InfoMan for each unique CCID.

Note: When using the action level of the CA Endevor InfoMan Interface, the user is allowed to wildcard the CCID field in CA Endevor SCM when requesting the following stand-alone actions in foreground and batch: ADD/UPDATE, MOVE, RETRIEVE, GENERATE, DELETE, TRANSFER, and ARCHIVE. When a CCID value is fully specified, a list is *not* produced. Processing proceeds normally, including processing specified by the interface. In addition, wildcarding CCIDs is not allowed when building SCL in packages.

Chapter 2: Installing the Interface

This section contains the following topics:

How CA Endevor InfoMan Interface Installation Works

To install the CA Endevor InfoMan Interface, you must do the following:

- Install basic InfoMan
- Modifying basic InfoMan to support the interface
- Link the CA Endevor SCM and InfoMan sides together
- Verify the installation

How Installation at Customized InfoMan Sites Works

If your InfoMan system has already been modified, the order of installation is as follows:

1. Obtain the data set names from Phase 1.
2. Refer to the appendix "InfoMan Information for Advanced Users (see page 141)," for considerations relevant to customized InfoMan shops or advanced users. The person in charge of customizing the InfoMan product needs to review the interface requirements in this appendix.
3. Perform the tasks listed in Phase 3.

How Installation at IIF Sites Works

If you use the InfoMan Integrated Facility (IIF), see the appendix "Installing the CA Endevor InfoMan Interface Under IIF (see page 147)," for installation instructions.

How to Install Basic InfoMan

The requirements in this section are basic InfoMan installation requirements that exist whether or not the interface is installed.

Note: CA Endevor InfoMan Interface requires that the IBM Information Management product, version 4.2 or higher, is installed at your site.

InfoMan setup involves the following.

1. Define all interface users to a class in InfoMan.
2. Make sure the Information Management Load Library data set is in LINKLST, or that you have the data set name available.
3. Prepare a session parameter (BLGSESxx) module to reflect the database, data dictionary, read panel, and RFT data sets that the interface uses when accessing the data base via the API.
4. Make sure that BLGISPFD, an InfoMan supplied ISPF panel, has been moved into your ISPPLIB concatenation.

Define Users to InfoMan Classes

Each person authorized to use the interface must be defined to a class in InfoMan.

Note: For more information, see your InfoMan documentation.

Prepare a Session Parameter Member

InfoMan requires a BLGSESxx module that points to the database, data dictionary, read panels, and RFT data set that the Interface uses when accessing the database via the API or native InfoMan.

This session parameter is well documented in the InfoMan install manuals. It is required of standard InfoMan. The naming standard is BLGSESxx where xx is the user's choice. This suffix is also required for a CA Endevor SCM start-up block assembly. Note that the same BLGSESxx member can be used for the interface as well as standard access to InfoMan.

The suffix used to name the actual BLGSESxx module is required in Phase 3. The rest of this install requires the data set names from the BLGSESxx module.

Note: The interface installation includes one modified Assisted Entry panel, which must be copied into your read panel data set concatenation. You can either overlay the IBM version of this data set, or provide a new read panel data set ahead of the standard IBM data set. Either way, the result is that the RPANEL0 label points to the data set used in the install.

Put Information Management Load Library Data Set in LINKLIST

The InfoMan load data set is used to install the interface.

Sometimes this library is part of LINKLIST, and sometimes it is not. If this data set is not in LINKLST, the person implementing the interface needs to know its name, and must put it into a STEPLIB in the concatenation.

Move BLGISPFD into ISPPLIB

Make sure that BLGISPFD, an InfoMan-supplied ISPF panel, has been moved into your ISPPLIB concatenation. See your InfoMan administrator if you have any questions about this step.

Summary of Basic InfoMan Installation

In this phase of the implementation you have built a vanilla version of InfoMan. At this point you should have installed InfoMan on your system, assigned users to an InfoMan class, and defined a BLGSESxx session parameter module. For future reference, record the following:

- BLGSESxx. Within BLGSESxx, you have defined the following:
  - Dictionary DD name:
  - RFTDS DS name:
  - RPANEL0 name:
- INFOMAN LOAD DS name:

How to Modify Basic InfoMan for the Interface

In this phase, you customize the base InfoMan system set up in Phase 1 by performing the following steps:

1. Copy interface report shells to the RFT data set identified in Phase 1. In this step, BC1JEI20 JCL is used.
2. Copy CA Endevor SCM version of BLG6CORQ into the RPANEL0 data set identified in Phase 1. In this step, BC1JEI15 JCL is used.
3. Install the dictionary update. In this step, BC1JEI30 JCL is used.
4. Create the standard IBM PIDT and PIPT tables. In this step, BC1JEI45 JCL is used.
5. Build a rule set for driving the API using the interface batch utility. In this step, BC1JEI00 JCL is used.
6. Assemble or copy the BLGSESxx module into your STEPLIB data set. In this step, BC1JEI05 JCL is used.

Copy Interface Report Shells into the RFT Data Set

Copy the interface report shells (RFTs) to your RFT data set. Do this using the BC1JEI20 member in *iprfx.iqual.CSIQJCL*. Before submitting this JCL, provide a job card, confirm that *iprfx.iqual* and uprfx.uqual are the correct data set qualifiers,and confirm that tdisk is a valid unit name.

This job copies the following report shells:

EINTACH0

Action Change Record Summary

EINTACH1

Action Change Record Detail

EINTAPB0

Action Problem Record Summary

EINTAPB1

Action Problem Record Detail

EINTPKG0

Package Parent Record Summary

EINTPKG1

Package/Activity Record Summary

EINTPKG2

Package/Activity Record Detail

Copy Modified BLG6CORQ into the RPANEL0 Data Set

The interface requires that a modified version of the Assisted Entry Panel (BLG6CORQ) be available in the RPANEL0 data set pointed to by the BLGSESxx session parameter member. The modification allows the BLG6CORQ panel to store the 16-character CA Endevor SCM package ID. This field has been flagged as a reserved field for the interface.

The modified panel is provided with the interface. Copy it into the RPANEL data set using the JCL in member BC1JEI15 in the iprfx.iqual.CSIQJCL data set.

This job uses the BLGUT6F utility to copy a PDS member that has been created from a InfoMan panel using BLGUT6.

This panel must reside above the IBM panel of the same name. The person who runs the job to copy the modified panel needs only the data set name. You recorded the RPANEL0 data set name in the worksheet at the end of Phase 1.

Copy Dictionary Update

The modified Assisted Entry Panel (BLG6CORQ) has an associated dictionary entry. This entry must be copied into the dictionary identified in the Phase 1 Summary. Run JCL job BC1JEI30 to install this dictionary entry.

Important! Run this job only if you are installing basic InfoMan. If your site has customized InfoMan, see the appendix "InfoMan Information for Advanced Users (see page 141)," for instructions on updating the dictionary entry.

BC1JEI30 JCL is shown next. Before submitting this JCL, provide a job card, and confirm that *iprfx.iqual* and uprfx.uqual are the correct data set qualifiers.

Create the Standard IBM PIDT and PIPT Tables

Create the PIDT and PIPT tables by executing InfoMan's BLGUT8 utility program. This step is necessary so that the updated dictionary entry is picked up. These tables are required by the interface to support batch and real-time interface execution, CCID list support and display services.

Use the BC1JEI45 member in iprfx.iqual.CSIQJCL. Before submitting this JCL, provide a job card and confirm that the *iprfx.iqual* and uprfx.uqual are the correct data set qualifiers.

Build a Rule Set

In addition to setting up InfoMan to support the interface, you need to build a rule set to drive the InfoMan API. You do this using the batch utility provided by the interface.

Before proceeding, review:

- Setting Up the E/INFO Table (see page 29)
- The samples provided in *iprfx.iqual.CSIQOPTN*. There are three samples provided:

EISCRIP1

Batch utility input for stand-alone actions, using InfoMan Change records.

EISCRIP2

Batch utility input for stand-alone actions, using InfoMan Problem records.

EISCRIP3

Batch utility input for package functions and package actions. Next, create a rule set. It is advisable to keep it simple at first. Do this by taking one or two actions or package functions and writing utility syntax input for them. Then run the batch utility to create an initial rule set. The JCL for executing the utility to build the rule set is in member BC1JEI00 of iprfx.iqual.CSIQJCL.

A successful run of the utility produces a load module called by a name you specify. This load module name is used in Phase 3. Assemble this load module into a STEPLIB data set.

Remember:

- To change what you use the interface for is a simple run of the utility. It requires no further InfoMan or CA Endevor SCM work to enhance the rule set.
- If you point to any PIDT tables other than the defaults, verify that they reside in the RFTDS data set pointed to in your BLGSESxx member.

The JCL for executing the utility to build the rule set is in member BC1JEI00 of iprfx.iqual.CSIQJCL. Before submitting this JCL, provide a job card, confirm that *iprfx.iqual* and uprfx.uqual are the correct data set qualifiers, confirm that tdisk is a valid unit name and, if necessary, change the size of the WORK1 or WORK2 data sets.

Copy the BLGSESxx Module to STEPLIB

Assemble or copy the BLGSESxx member into your STEPLIB data set. See iprfx.iqual.CSIQJCL(BC1JEI05) for sample JCL.

Before submitting this JCL, provide a job card, confirm that *iprfx.iqual* and uprfx.uqual are the correct data set qualifiers, confirm that tdisk is a valid unit name,and replace xx in the string BLGSESxx with a session identifier.

How to Connect the Interface

You must complete the tasks described for Phase 1 and Phase 2 before starting the tasks described in this section.

This phase of the installation process involves setting up the InfoMan and CA Endevor SCM sides of the interface to work together, as described in the following.

1. Modify the @EINFO macro.
2. Assemble and link BC1TEI90 into a LINKLIST or authorized library.
3. Assemble C1DEFLTS with InfoMan password.
4. Make sure that required panels are in the ISPPLIB data set.
5. Make sure that members INFO01 and PKMR02 are in your ISPMLIB data set.
6. Edit C1SB3000 skeleton JCL.

Modify the @EINFO Macro

Various parts of the interface access the user-assembled module BC1TEI90 to obtain user parameter information. BC1TEI90 is a load module used during exits 5, 2, 3, and 7 for start-up parameters, during display services, and during batch execution. The user must assemble this load module. Once assembled, this module should be linked into a LINKLIST or authorized library.

The @EINFO macro, supplied with the interface, provides input to BC1TEI90. The macro can be found embedded in member BC1JEI10 in iprfx.iqual.CSIQJCL. Edit this macro based on the information contained in the Phase 1 checklist and your site standards.

The following is the @EINFO macro:

```
@EINFO ENTRY=START,
     BLGSESS=00,
     APISESID=USERID,
     EPIDT=ENDEVOR,
     INVCLASS=MASTER,
     WAITTIME=300,
     RECTYPE=CHANGE,
@EINFO ENTRY=END
```

The entries in this macro are described next.

ENTRY=START

Indicates the beginning of the BC1TEI90 table.

BLGSES=

InfoMan looks for session parameters in a module named BLGSESxx, where xx is a two-character identifier for the module. This module is described in the InfoMan documentation. You recorded the components of this module during Phase 1 of this installation, as well as the two-character identifier for the module. When modifying the @EINFO macro supplied with the interface, replace the default value 00 with the value you specified in Phase 1.

The default is 00. This is the same as standard InfoMan.

APISESID=

The API requires a session ID during initialization. This session can be called USERID or can be a predefined InfoMan API ID.

The default is USERID. Leaving this default allows the same start-up module to be used by many people, with the interface substituting the user ID of the caller for the literal USERID.

If you include this keyword, you must provide a value.

EPIDT=

This keyword contains the name of the assembled E/INFO table produced by the batch utility. The default is Endevor.

If you include this keyword, you must provide a value.

INVCLASS=

InfoMan has its own security system. It is broken into classes with privileges. Every user or API ID is assigned to at least one CLASS. The API requires a class name.

The default is MASTER.

WAITTIME=

The API allows the application to define how long it will wait for a response from the InfoMan API server. If the user does not use this keyword, the default will be 300 seconds.

RECTYPE=

Users must select one kind of InfoMan record for recording information about stand-alone actions. They can use either Change or Problem records. Display services also need this information.

The default is CHANGE. If coded, it cannot be blank.

ENTRY=END

Required. This statement denotes the end of the BC1TEI90 table.

Assemble and Link BC1TEI90

Assemble and link BC1TEI90 using the JCL in member BC1JEI10 in iprfx.iqual.CSIQJCL. Before submitting this JCL, provide a job card, confirm that *iprfx.iqual* and *uprfx.uqual* are the correct data set qualifiers, and confirm that tdisk is a valid unit name.

```
//*
//*********************************************************************
//*  THE INPUT SHOULD BE ASSEMBLED AND LINKED INTO A STEPLIB DATASET. *
//*  PLEASE REFER TO THE INTERFACE CHAPTER COVERING IMPLEMENTATION AND*
//*  INSTALLATION.                                                    *
//*                                                                   *
//*********************************************************************
//*
//STEP2    EXEC PGM=IEWL,PARM='LIST,NCAL,RENT,XREF,SIZE=(256K,64K)',
//             COND=(0,NE)
//SYSLIN   DD  DISP=(OLD,DELETE),DSN=&&SYSLIN.
//SYSLMOD  DD  DISP=SHR,DSN=uprfx.uqual.LOADLIB(BC1TEI90)
//SYSUT1   DD  UNIT=tdisk,SPACE=(CYL,(5,3))
//SYSPRINT DD  SYSOUT=*
```

Reassemble C1DEFLTS with the InfoMan Password

Include the InfoMan password in your C1DEFLTS table. The password parameter is in the TYPE=MAIN section of the defaults table, as shown below.

Make Sure Required Panels Are in ISPPLIB

Make sure that the panels listed next reside in your ISPPLIB. These panels can be found in *iprfx.iqual.CSIQPENU*.

C1EILIST, CITILIST

CCID list panel and its associated tutorial panel.

C1SEIBRW, CITEIBRW

InfoMan record display for packages and stand-alone actions and its associated tutorial panel.

C1SP1000, C1SP2000, C1SP3000, C1SP4000, C1SP5000, C1SP6000, C1SP7000

Package panels with capability to display correlation data.

CITP1000, CITP2000, CITP3000, CITP4000, CITP5000, CITP6000, CITP7000

Tutorials for package panels with capability to display correlation data.

Make Sure INFO01, PKEX21, and PKMR02 Are in ISPMLIB

Members INFO01, PKEX21. and PKMR02 contain interface messages, and must reside in your ISPMLIB. These members can be found in *iprfx.iqual*.CSIQMENU.

Edit C1SB3000 Skeleton JCL

Member C1SB3000 is used to submit CA Endevor SCM package and batch processing requests. The member can be found in *iprfx.iqual*.ISPSLIB. Before submitting this JCL, confirm that *iprfx.iqual* are the correct data set qualifiers, and confirm that tdisk is a valid unit name.

How to Verify the Installation

The following list shows the members that should reside in the designated data sets for the interface to function as designed. If any of them are not there, the interface will not work as designed.

CSIQAUTH/CSIQAUTU

Load modules, as well as the following:

- C1BMEI00
- C1DEFLTS table. See Step 3. Reassemble C1DEFLTS with the InfoMan Password.
- C1SM1000
- BC1TEI90, the module containing user parameters, which you assembled in Step 2. Assemble and Link BC1TEI90.
- E/INFO table. This is the rule set that you build using the batch utility. See Step 5. Build a Rule Set.
- BLGSESxx, the module containing the InfoMan session parameters. See Step 2. Prepare a Session Parameter Member.

CONLIB

The CA Endevor SCM load modules installed from *iprfx.iqual*.CSIQLOAD. Make sure to save the current copy of C1BEXITS before installing the new interface. This allows you to go back to the old interface during testing.

RFTDS

EINTACH0, EINTACH1, EINTAPB0, EINTAPB1, EINTPKG0, EINTPKG1, EINTPKG2.

PIDT and PIPT tables. See the list in the next section. These panels are found in iprfx.iqual.CSIQOPTN.

BLGISPFD, the InfoMan supplied ISPF panel.

RPANEL0

BLG6CORQ, the modified version of the Assisted Entry panel. See Step 3. Copy Dictionary Update.

DDICT

index RNCC/CCV15. This is the modified dictionary entry associated with the modified Assisted Entry Panel. See Step 4. Create the Standard IBM PIDT and PIPT Tables.

iprfx.iqual. ISPPLIB

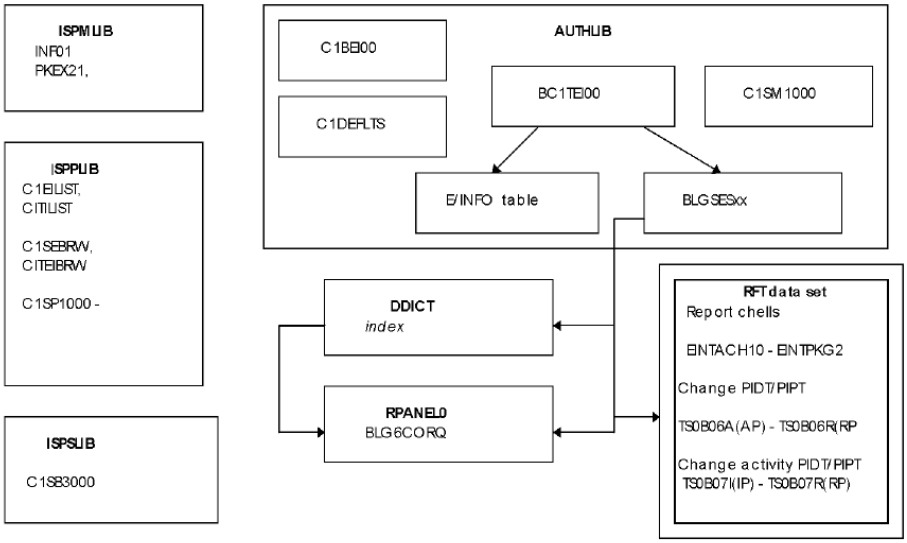
C1EILIST, C1SEIBRW, C1SP1000 — C1SP7000 and the associated tutorials (CITILIST, CITEIBRW, and CITP1000-CITP7000.

iprfx.iqual. ISPMLIB

INFO01, PKEX21, PKMR02.

iprfx.iqual. ISPSLIB

C1SB3000. The following shows how these modules tie the interface together, and indicates the important members that should be at each location.



As previously illustrated, you can see how the modules tie the interface together and which important members should be at each location.

Required PIDT and PIPT Tables

For the interface to work correctly the following PIDT and PIPT tables must be in the RFT data set pointed to by the BLGSESxx module.

There are three categories of PIDT/PIPT tables: Change, Change Activity, and Problem. These tables are listed next, and can be found in iprfx.iqual.CSIQOPTN.

Change Record PIDT and PIPT Tables

The following PIDT/PIPT tables related to Change records must be installed.

TS0B06A

Change record, Add, PIDT

TS0B06AP

Change record, Add, PIPT

TS0B06I

Change record, Inquiry, PIDT

TS0B06IP

Change record, Inquiry, PIPT

TS0B06C

Change record, Create, PIDT

TS0B06CP

Change record, Create, PIPT

TS0B06U

Change record, Update, PIDT

TS0B06UP

Change record, Update, PIPT

TS0B06R

Change record, Retrieve, PIDT

TS0B06RP

Change record, Retrieve, PIPT

Change Activity PIDT and PIPT Tables

The following PIDT/PIPT tables related to Change Activity records must be installed.

TS0B07I

Change activity record, Inquiry, PIDT

TS0B07IP

Change activity record, Inquiry, PIPT

TS0B07C

Change activity record, Create, PIDT

TS0B07CP

Change activity record, Create, PIPT

TS0B07U

Change activity record, Update, PIDT

TS0B07UP

Change activity record, Update, PIPT

TS0B07R

Change activity record, Retrieve, PIDT

TS0B07RP

Change activity record, Retrieve, PIPT

Problem Record PIDT and PIPT Tables

The following PIDT/PIPT tables related to Problem records must be installed.

TS0032I

Problem record, Inquiry, PIDT

TS0032IP

Problem record, Inquiry, PIPT

TS0032C

Problem record, Create, PIDT

TS0032CP

Problem record, Create, PIPT

TS0032U

Problem record, Update, PIDT

TS0032UP

Problem record, Update, PIPT

TS0032R

Problem record, Retrieve, PIDT

TS0032RP

Problem record, Retrieve, PIPT

Chapter 3: Setting Up the CA Endevor SCM INFO Table

This section contains the following topics:

The CA Endevor SCM Side of the Interface

CA Endevor SCM communicates with the InfoMan API through exit points, using a control block referred to here as the CA Endevor SCM INFO table (E/INFO table).

The purpose of the E/INFO table is to provide a flexible, easy way to implement site InfoMan policy through the InfoMan API. In particular, the E/INFO table allows the user to:

- Utilize custom PIDT tables to provide logical views of InfoMan records.
- Explicitly specify the exit points where actions are to be performed.
- Establish criteria on the CA Endevor SCM or the InfoMan side for performing an action.
- Specify the CA Endevor SCM information to be recorded in InfoMan, and the InfoMan fields where the information is to be recorded.
- Identify InfoMan actions to be performed (Inquiry, Retrieve, Create, Update), and the PIDT table needed for each action.

You build the E/INFO table during interface implementation using a batch utility provided with the interface. This utility assembles the table into a load module that is invoked at the exit points defined in the table.

When the table is invoked, CA Endevor SCM builds a request for the InfoMan API using the information in the table. The InfoMan API executes the requested actions against the InfoMan database, using the requested PIDT tables.

This chapter describes the syntax for the E/INFO table.

How to Implement the CA Endevor SCM Side

Before writing the table definition statement, complete the following steps:

1. Determine what kind of CA Endevor SCM information you want to store in InfoMan: package, package action, or stand-alone action information.
2. Be familiar with the necessary CA Endevor SCM exit points and exit control blocks for the kinds of information you plan to track.

   Note: For more information, see the *Administration Guide*.
3. Know which control block fields for the exit points you wish to utilize contain information eligible for storage in InfoMan.
4. Decide which of the available control block fields you want to track in InfoMan.
5. Decide InfoMan fields you want to use to store the CA Endevor SCM information.

After writing the table definition statement, submit it to the CA Endevor SCM batch utility.

CA Endevor SCM INFO Table Syntax

Here is an example of the syntax used to produce an E/INFO table. The rest of the chapter discusses this example.

```
TABLE NAME (CASCMMF);

CTRL;
     IACT (TS0B06I);         /* Inquiry Change for actions    */
     RACT (TS0B06R);         /* Retrieve Change for actions   */
     CACT (TS0B06C);         /* Create Change for actions     */
     UACT (TS0B06U);         /* Update Change for actions     */
```

```
ECTRL;
     USE(ADD,BEFORE,STANDALONE);
         NOTFOUND (CREATE,ADD0010);
         FOUND        (ADD0020);
         LABEL ADD0010;
             CRITERIA;
             ECRITERIA;
                 POPULATE;
                   FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
                   FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
                   FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
                     VALUE('API CREATED CHG RECORD');
                 EPOPULATE;
             RC (0);
         ELABEL;
         LABEL ADD0020;
             CRITERIA;
               FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
             ECRITERIA;
             POPULATE;
             EPOPULATE;
         CPASS ADDPASS;
         CFAIL ADDFAIL;
         ELABEL;
         LABEL ADDPASS;
             RC (0);
         ELABEL;
         LABEL ADDFAIL;
             RC (8);
         ELABEL;
     EUSE;
     USE(ADD,AFTER,STANDALONE);
         FOUND (ADD0010);
         NOTFOUND (RETURN);
         LABEL ADD0010;
             CRITERIA;
             ECRITERIA;
             POPULATE;
                 FIELD PANEL(BLG0C010) INDEX(S0E01)
                 TEXT('PROCESSING COMPLETED FOR ELEMENT =EV2ELEMT ')    -
                  ('ACTION: =ACTION   EXECUTED BY: =USER '  ) -
                  ('ON =DATE =TIME');
             EPOPULATE;
         ELABEL;
     EUSE;
ETABLE;
```

Table Syntax Components

The major components of the E/INFO table syntax are shown below.

Table statement

One per table

Starts, names, and ends the table.

Control (CTRL) block

One per table

Identifies the InfoMan PIDT tables to be used by the requests produced by the E/INFO table.

Use blocks

One or more per table

Identifies the exit points where the table is invoked and associate processing logic with each exit point.

Label blocks

One or more per Use block

Specify processing for an exit point.

Criteria blocks

One or more per Use block

Establish true/false conditions. When there is a true condition, the interface populates InfoMan records with CA Endevor SCM information.

Populate blocks

One or more per Use block

Identifies CA Endevor SCM information to be recorded in InfoMan. The statements that can be coded for each block are described later in this chapter.

Syntax Diagram Conventions

The following conventions are used in the syntax diagrams in this chapter:

- User-supplied variables are presented in italics. For example:

  ```
  TABLE NAME (name);
  ```

- The acceptable values for the variables are supplied in the discussion of the syntax.

- Optional syntax components are enclosed in brackets. For example, in the syntax below the components STANDALONE and PROBLEM are (Optional)

  ```
  USE(function,{BEFORE|AFTER|ACTIVITY}[,STANDALONE][,PROBLEM])
  ```

- Required syntax components for which there is more than one acceptable value are shown enclosed in braces, with the acceptable values separated by vertical lines. For example, in the syntax below the second argument between parentheses must be BEFORE, AFTER, or ACTIVITY.

  ```
  USE(function,{BEFORE|AFTER|ACTIVITY}[,STANDALONE][,PROBLEM])
  ```

The TABLE Statement

The TABLE statement starts a table, names the table, and ends the table.

The syntax is:

```
TABLE NAME (name);
.

ETABLE;
```

The following are descriptions of syntax components:

TABLE NAME

(Required) This must be the first statement in the E/INFO table syntax.

name

(Required) A 1- to 8-alphanumeric character name for the table. This name is also used as the output member name of the load module that interacts with the InfoMan API.

Parentheses on the name are (Optional) The name must be followed by a semicolon.

ETABLE

(Required) This must be the final statement in the E/INFO table syntax. It must be followed by a semicolon.

Example:

```
TABLE NAME (NDVR);
.
ETABLE;
```

This example builds an E/INFO table called NDVR.

The Control Block

The Control (CTRL) block specifies the:

- InfoMan functions that the table may request.
- CA Endevor SCM entities affected by the table.
- PIDT tables that must be available to the E/INFO table.

The CTRL block must follow the TABLE statement. There must be a statement in this block for each function that you want to perform against the InfoMan database. The statements in this block must identify all the PIDT tables required by the requested functions.

Note: Users of standard InfoMan should always use the standard CTRL statements included in the samples.

Control Block Syntax

The syntax for the Control block is:

```
CTRL;

     InfoMan action CA Endevor SCM entity (PIDT name)
     ...
ECTRL;
```

The CTRL Statement

The CTRL keyword is (Required) It must be the first line in the control statement and must be followed by a semicolon. The parentheses surrounding the PIDT name are optional; however, the PIDT name must be followed by a semi-colon.

InfoMan Action

An InfoMan action is (Required) There must be at least one InfoMan action specified in a control statement. Acceptable values are

- I (Inquiry)
- R (Retrieve)
- C (Change)
- U (Update)
- A (Add)

Use these actions as shown:

Inquiry (I)

Check to see if an InfoMan record exists.

Retrieve (R)

Test a field in an InfoMan record for a specific value.

Create (C)

Create records in InfoMan based on criteria in the E/INFO table.

Update (U)

Update existing records in InfoMan.

Add (A)

Use the InfoMan Change Activity record type to store information about package actions. The InfoMan action appears together with a CA Endevor SCM entity to which it applies. See the next section for a description of CA Endevor SCM entities.

CA Endevor SCM Entity

A CA Endevor SCM entity is required and specifies an entity for which you want to track information. Acceptable values are ACT (stand-alone actions), PKG (packages), and PKGA (package actions). For a table listing possible combinations of InfoMan actions and CA Endevor SCM entities, see Possible Requests.

PIDT Name

A PIDT name is (Required) Each request clause must identify an InfoMan PIDT table. The InfoMan API uses this table to log CA Endevor SCM information in InfoMan.

Parentheses are (Optional) The PIDT table name must be followed by a semicolon.

ECTRL Statement

An ECTRL statement is required and must be the final line in the control statement. It must be followed by a semicolon.

Possible Requests

The following API requests may be defined in this block.

AACT

Add activity record creations.

Package action exit points, Create Activity, and one or more Populate blocks.

CACT

Action Change or Problem record creations.

Action exit points and one or more populate blocks.

CPKG

Package Change record creations.

Package exit points and one or more Populate blocks.

CPKGA

Package Activity record creations.

Package action exit points and one or more Populate blocks.

IACT

Action Change or Problem record inquiries.

Stand-alone action exit points.

IPKG

Package exit points.

IPKGA

Package action exit points.

Create activity.

RACT

Action Change or Problem record retrievals.

Action exit points and one or more Criteria blocks.

RPKG

Package Change record retrievals.

Package exit points and one or more Criteria blocks.

RPKGA

Package Activity record retrievals.

Package action exit points and one or more Criteria blocks.

UACT

Action Change or Problem record updates.

Action exit points, the STANDALONE keyword and one or more Populate blocks.

UPKG

Package Change record updates.

Package exit points and one or more Populate blocks.

UPKGA

Package Activity record updates.

Package action exit points and one or more Populate blocks.

Create Activity and one or more Populate blocks.

CTRL Statement Examples

To store and maintain stand-alone action information in InfoMan, use this CTRL statement:

```
CTRL;
     IACT (TS0B06I);   /* Inquiry Change for actions    */
     RACT (TS0B06R);   /* Retrieve Change for actions   */
     CACT (TS0B06C);   /* Create Change for actions     */
     UACT (TS0B06U);   /* Update Change for actions     */
ECTRL;
```

This statement identifies four Change record PIDT tables (TS0B06I, TS0B06R, TS0B06C, TS0B06U), and indicates that the table may request any of the available API actions (Inquiry, Retrieve, Create, or Update).

To store and maintain package and package action information in InfoMan, use this CTRL statement:

```
CTRL;
     IPKG  (TS0B06I);  /* Inquiry Change table for packages           */
     IPKGA (TS0B07I);  /* Inquiry Activity table for package actions  */
     RPKG  (TS0B06R);  /* Retrieve Change table for packages          */
     RPKGA (TS0B07R);  /* Retrieve Activity table for package actions */
     CPKG  (TS0B06C);  /* Change Change table for packages            */
     CPKGA (TS0B07C);  /* Change Activity table for package actions   */
     UPKG  (TS0B06U);  /* Update Change table for packages            */
     UPKGA (TS0B07U);  /* Update Activity table for package actions   */
ECTRL;
```

This statement tells the InfoMan API that this E/INFO table may request access to any of four Change record PIDT tables (TS0B06I, TS0B06R, TS0B06C, TS0B06U) for storing package information, and four Activity record tables (TS0B07I, TS0B07R, TS0B07C, TS0B07U) for storing package action information.

Use Blocks

A Use block identifies one or more exit points where the CA Endevor InfoMan Interface is to be invoked. A table definition can have multiple Use blocks.

All exit points at which you want to communicate with InfoMan must appear in a Use block. A single USE block can refer to one, or more than one exit point. If a Use block refers to more than one exit point, all exit points in the Use block must:

- Be of the same kind. For example, a Use block can refer to multiple stand-alone action exit points or multiple package exit points, but cannot refer to both stand-alone action exit points and package exit points.
- Utilize the same control blocks. Actions that can appear in the same Use block are:
  - ADD, UPDATE, RESTORE
  - RETRIEVE, ARCHIVE
  - TRANSFER, MOVE
  - GENERATE, DELETE
  - BACKIN, BACKOUT, CAST, COMMIT, EXECUTE, RESET, REVIEW, SHIP

The following actions must have their own Use block:

- CREATE Activity
- CREATE Before
- CREATE After

- PDELETE Before
- PDELETE After
- GENPKGID

Each Use block has the following components:

- A USE statement to identify one or more exit points where the E/INFO table is to be invoked. Each USE statement has one or more required CTRL block statements.
- A NOTFOUND statement to indicate what to do if an InfoMan record does not exist for this exit point, and which points to a Label block that specifies processing.
- A FOUND statement to indicate what to do if an InfoMan record exists for this exit point, and which points to a Label block that specifies processing.
- One or more Label blocks, to specify the processing associated with the exit points named in the USE statement.
- A EUSE statement to mark the end of the Use block.

The USE Statement

The USE statement (required) is the first statement in a Use block, and identifies the exit point or points referred to by the Use block. The USE statement must be the first word in the Use block.

The syntax for referring to a single exit point is:

```
USE(action,{BEFORE|AFTER|ACTIVITY}[,STANDALONE][,PROBLEM]);
```

The syntax for referring to multiple exit points is:

```
USE((action,{BEFORE|AFTER|ACTIVITY}[,STANDALONE][,PROBLEM])
       ...              (action,{BEFORE}[  .,STANDALONE][,PROBLEM]));
```

The Action Variable

This variable identifies the action for the Use block. The action along with the subparameter Before/After/Activity, uniquely identifies a CA Endevor SCM exit point. Valid combinations are:

Action	Subparameter
Backin	Before/After
Backout	Before/After
Cast	Before/After
Commit	Before/After
Confirm	After
Create	Before/After
Create	Activity. This exit point builds activity records for package actions at after-cast time.
Pdelete	Before/After (Package Delete)
Execute	Before/After
GENPKGID	Before/After
Modify	Before/After
Reset	Before/After
Review	Before/After
Ship	Before/After
Add	Before/After
Update	Before/After
Restore	Before/After
Retrieve	Before/After
Archive	Before/After
Generate	Before/After
Delete	Before/After
Move	Before/After
Transfer	Before/After

The STANDALONE Subparameter

This subparameter is only valid with action functions. When used, it identifies actions running outside of the package environment.

The PROBLEM Subparameter

This subparameter can only be used with stand-alone actions. When used, it means that a Problem record will be used instead of a Change record when recording information in the InfoMan database.

USE Statement Syntax Examples

Here is an example of USE statement syntax for a single exit point:

```
USE(ADD, BEFORE, STANDALONE);
```

This example indicates that the Use block refers to a before-action exit for ADD actions, and that the ADD actions are performed in stand-alone mode (not as part of packages).

Here is an example of USE statement syntax for multiple exit points:

```
USE    ((ADD, AFTER),
       (UPDATE, AFTER),
       (RESTORE, AFTER));
```

This example indicates that the Use block applies to the after-exit point for the named actions, when the actions are included in a package.

The NOTFOUND Statement

NOTFOUND and FOUND statements must follow the USE statement. Each statement can appear only once for a Use block.

Once an exit point is driven, a search is made to locate the InfoMan record from the database. This statement instructs the interface how to proceed when the InfoMan record has not been created yet. This statement is required for:

- All stand-alone action exit points.
- The following package exit points: Create, Modify, Cast, Delete.

This statement is not valid when used with the package exit points for Backout, Backin, Commit, Confirm, Create Activity, Execute, Reset, Review, and Ship.

Statement syntax is:

```
NOTFOUND ({[CREATE,] label | RETURN});
```

NOTFOUND

(Required) NOTFOUND must be the first word in this statement.

CREATE

(Optional) The CREATE keyword tells the interface to create one of the following in InfoMan:

A package change record. If you want to do this, the CTRL block must contain a CPKG statement.

A change activity record for a package action. If you want to do this, the CTRL block must contain CPKGA and AACT statements.

A change or problem record for a stand-alone action. If you want to do this, the CTRL block must contain a CACT statement.

The criteria for populating the new InfoMan record are contained in the block identified by the label parameter.

If you want to update existing InfoMan records, omit CREATE from the NOTFOUND statement, and make sure that UPKG, UPKGA, or UACT statements are in the CTRL block.

label

(Optional) This identifies the label name where execution resumes when a NOTFOUND condition occurs. Required if CREATE is the first parameter.

RETURN

(Optional) Instructs the interface to return to CA Endevor SCM. Examples of NOTFOUND statements appear in the section on FOUND statements.

The FOUND Statement

This statement directs the logic process when the InfoMan record has been found.

```
FOUND ({[DELETE,] label | DELETE,RETURN | RETURN}  );
```

The following are descriptions of syntax components:

FOUND

(Required) FOUND must be the first word in this statement.

DELETE

(Optional) This parameter is only valid on the Package Delete (PDELETE) After exit point. This option deletes the Package Change record along with all its corresponding activity records.

If you use this parameter, make sure that PDELETE is the first parameter in the USE statement.

label

(Optional) This identifies the label name where execution resumes when a FOUND condition occurs. Required if DELETE is the first parameter.

RETURN

(Optional) Instructs the interface to return to CA Endevor SCM.

NOTFOUND/FOUND Statement Examples

Here is an example of NOTFOUND/FOUND statements

```
NOTFOUND (CREATE,CMD0010);
FOUND    (CMD0020);
```

This example indicates that if the interface does not find a record on the InfoMan side, it is to create that record. The information needed to create the record can be found later in the table at label CMD0010.

The example also indicates that if the record is found in InfoMan, the interface should execute the processing specified in label CMD0020.

Here is a second example of NOTFOUND/FOUND statements:

```
NOTFOUND (ERR0010);
FOUND (RETURN);      /* GREAT —— JUST RETURN      */
```

This example indicates that if the record is not found, the interface is to execute the processing specified in label ERR0010, and that if the record is found, the interface should return to CA Endevor SCM.

The EUSE Statement

Each USE statement must have a corresponding EUSE statement as the last statement in a Use block. Its syntax is:

```
EUSE;
```

Example

This example shows the overall structure of a Use block:

```
USE(GENERATE, AFTER, STANDALONE);
    FOUND (GEN0010);
    NOTFOUND (RETURN);
    LABEL GEN0010;
        CRITERIA;
        ECRITERIA;
        POPULATE;
          FIELD PANEL(BLG0C010) INDEX(S0E01)
                TEXT ('Action processing completed rc = =ACTRC')
                TEXT ('ELEMENT: =EV1ENV / =EV1SYS / =EV1SUB ') -
                          ('/ =EV1TYP / =EV1STAGE / =EV1ELM')
                TEXT ('ACTION:  =ACTION  USER: =USER  DATE: =DATE ') -
                          ('TIME: =TIME');
        EPOPULATE;
    ELABEL;
EUSE;
```

Label Blocks

Label blocks specify processing that is to occur related to the exit point in a Use block. A Label block must be within a Use block. Multiple Label blocks are allowed per exit point.

Label blocks must contain:

- A LABEL statement
- An ELABEL statement

The Label block can also contain:

- CPASS/CFAIL statements
- TSP statements
- Return code (RC) statements
- Message (MSG) statements

- Populate blocks
- Criteria blocks

The LABEL statement marks the beginning of the block. All statements within the Label block are processed. Once the ELABEL statement is reached, control is returned to CA Endevor SCM unless a CPASS/CFAIL statement is included in the block.

The LABEL Statement

The LABEL statement indicates the start of a Label block, and must also contain the name of that block. The syntax is:

```
LABEL name;
```

The following are descriptions of syntax components:

LABEL

(Required) LABEL must be the first word in a Label block.

name

(Required) A 1- to 8-character alphanumeric name for the label. The name must be followed by a semicolon.

Example

This example shows the use of a comment with the LABEL statement:

```
LABEL GEN0010;  /*  Change Record not found:  Create one. */
```

The ELABEL Statement

The ELABEL statement must be the last statement in a Label block. The syntax is:

```
ELABEL;
```

The semicolon at the end of the statement is required.

CPASS/CFAIL Statements

Use CPASS/CFAIL statements to pass control to another Label block when the current block's ELABEL statement is reached, based on the conditions set by the Criteria block. The syntax is:

```
{CPASS|CFAIL} label-name;
```

The following are descriptions of syntax components:

CPASS

Indicates a true condition.

CFAIL

Indicates a false condition.

label name

The up to eight character name of the label where processing resumes when the true or false condition has been met. There can be one CPASS/CFAIL statement per Label block. The CPASS and CFAIL statement may exist anywhere in the block, but the condition is only tested and the branch occurs just prior to the ELABEL.

In order to code these statements, a Criteria and a Populate block must be present in the Label block. CPASS represents a true condition and CFAIL represents a false condition. CPASS and CFAIL may be coded independently.

Coding a null Criteria block sets a CPASS condition. However, if you want to pass control to another Label block, you must still code the CPASS statement, specifying the Label block to which you want to pass control.

Example (CFAIL Statement)

Here is an example of the use of CFAIL:

```
LABEL ARC0020;       /*  Change Record found  */
   RC (0);
   CRITERIA;
     FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
   ECRITERIA;
       POPULATE;
       EPOPULATE;
   CFAIL ARC0022;
ELABEL;
```

In this example, a PIDT field in InfoMan would be checked for the value OPEN. The CFAIL statement tells CA Endevor SCM to execute the processing specified in Label block ARC0022 if the field does not contain the value OPEN.

Example (CPASS Statement)

Here is an example of the use of CPASS:

```
LABEL CST0010;       /* CHANGE RECORD NOT FOUND:  CREATE ONE. */
   CRITERIA;
   ECRITERIA;
        POPULATE;
             /*************************************
              * PECBUSER:  CURRENT USER ID         *
              *************************************/
           FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=PECBUSER);
           FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
           FIELD PANEL(BLG0C010) INDEX(S0E01)
           VALUE('E/INFO CREATED CHANGE RECORD');
        EPOPULATE;
    CPASS CST0020;
ELABEL;
```

In this example, the interface first executes the processing in this Label block to create a change record in InfoMan. The CPASS statement instructs CA Endevor SCM to execute the processing specified in Label block CST0020, because a null Criteria block sets a true condition.

Example (CPASS and CFAIL Statements Together)

You can use CPASS and CFAIL statements together when you want to test a CA Endevor SCM or an InfoMan field for the presence of a value, then execute one Label block if the value exists, and another Label block if the value does not exist.

```
LABEL CRT0020;       /* LOOKING FOR ACTION: ADD            */
  CRITERIA;
     FIELD (=PACTACTN) VALUE('ADD     ');
  ECRITERIA;
     POPULATE;
     EPOPULATE;
  CPASS GRP0010;
  CFAIL CRT0025;
ELABEL;
```

In this example, CA Endevor SCM tests the PACTACTN field for the presence of the value ADD. The CPASS statement instructs CA Endevor SCM to continue processing at Label block GRP0010 if ADD is the value in this field. The CFAIL statement instructs CA Endevor SCM to resume processing at Label block CRT0025 if ADD is not the value in the field.

TSP Statements

There can be one TSP statement per Label block. A TSP statement must be placed before any Criteria or Populate blocks. The syntax is:

```
TSP;
```

A TSP statement instructs the InfoMan API to invoke a TSP. The user is responsible for developing this TSP. The TSP is invoked within the InfoMan TSP driver BLGAPI00. The user must update the InfoMan TSP driver to include his/her TSP by including a LINK statement for transaction 111. For more information regarding TSPs and the InfoMan see the IBM Tivoli Information Management Application Program Interface Guide SC34-4592-00.

A TSP statement can appear only in the initial Label block of a chain of Label blocks. TSP statements cannot be coded in a block to which control has been passed by a CPASS or CFAIL statement.

Example (Valid TSP Statements)

The TSP statements in this example are allowed, because LABEL (ABC) and LABEL (CDE) each start a chain of Label blocks and therefore can each contain a TSP statement.

```
NOTFOUND (ABC);
FOUND (CDE);
.
LABEL ABC;
TSP;
.
CPASS FGH;
ELABEL;
LABEL CDE;
TSP;
.
CPASS KLM;
ELABEL;
```

Example (Invalid TSP Statements)

The TSP statement in LABEL (CDE) below is not allowed, because LABEL (CDE) is the second block in a chain, pointed to by the CFAIL statement in LABEL (ABC).

```
NOTFOUND (ABC);
FOUND (CDE);
.
LABEL ABC;
TSP;
.
CFAIL CDE;
ELABEL;
LABEL CDE;
TSP;
.
CPASS KLM;
ELABEL;
```

Comments

You can put comments anywhere within the E/INFO table.

A comment must begin with the character string /* and must end with the character string */. Comments can span multiple lines. The terminating character string, */, must be on the same line.

Comments must appear after the semicolon that ends a statement.

Example (Singe-line Comments)

Here is an example of single-line comments:

```
 CTRL;
     IACT (TS0B06I);    /* INQUIRY CHANGE FOR ACTIONS   */
     RACT (TS0B06R);    /* RETRIEVE CHANGE FOR ACTIONS  */
     CACT (TS0B06C);    /* CREATE CHANGE FOR ACTIONS    */
     UACT (TS0B06U);    /* UPDATE CHANGE FOR ACTIONS    */
   ECTRL;
```

Example (Multi-line Comments)

Here is an example of a comment that spans multiple lines:

```
POPULATE;
   /****************************************
   * PREQCOMM:  PACKAGE COMMENT             *
   * PREQEWSD:  EXECUTION WINDOW START DATE  *
   * PREQEWST:  EXECUTION WINDOW START TIME  *
   * PREQEWED:  EXECUTION WINDOW END DATE    *
   * PREQEWET:  EXECUTION WINDOW END TIME    *
   * PECBUSER:  CURRENT USER ID              *
   * PECBFNNM:  PACKAGE FUNCTION             *
   * PECBBANM:  BEFORE OR AFTER              *
   * PECBNDRC:  NDVR HIGH RETURN CODE        *
   * PHDRSTAT:  PACKAGE STATUS               *
   * PHDRCRD:   PACKAGE CREATION DATE        *
   * PHDRCRT:   PACKAGE CREATION TIME        *
   ****************************************/
```

In this example, the starting string (/*) and the ending string (*/) of the comment are shown in bold.

Example (Invalid Comments)

The comment below is invalid because it appears between the end of the statement and the semicolon that ends the statement.

```
CTRL;
     IACT (TS0B06I)  /* INQUIRY CHANGE FOR ACTIONS   */; T
```

The following is the correct way to code this comment:

```
CTRL;
     IACT (TS0B06I);  /* INQUIRY CHANGE FOR ACTIONS   */
```

RC Statements

Use RC statements to specify a return code back to CA Endevor SCM. This statement may appear anywhere within a Label block. The syntax is:

```
     RC (return code);
```

Acceptable return code values are 0 (success) or 8 (failure). If the interface returns an “8” on a before-action or before-package exit, CA Endevor SCM aborts the function.

Multiple RC statements are allowed within the same block. When multiple RC statements are used, the interface returns the highest value.

Example (RC Statement at the end of a Label block)

Here is an example of an RC statement at the end of a Label block:

```
LABEL MOV0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
   CRITERIA;
    ECRITERIA;
       POPULATE;
          FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
          FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
          FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
             VALUE('API CREATE CHG RECORD')
       EPOPULATE;
       RC(0);
ELABEL;
```

Here is an example of an RC statement at the start of a Label block:

```
LABEL MOV0020;       /* CHANGE RECORD FOUND */
       RC (0);
    CRITERIA;
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
    ECRITERIA;
       POPULATE;
       EPOPULATE;
    CFAIL MOV0022;
ELABEL;
```

In both cases, the RC statement instructs the interface to send a return code of 0 back to CA Endevor SCM when it has completed the processing specified in the Label block.

MSG Statements

Use message (MSG) statements to include a message in the CA Endevor SCM Execution Report. This statement may appear anywhere within a Label block. Multiple occurrences are allowed.

The syntax is:

```
MSG ('text');
```

The text must be enclosed in single quotes. The maximum length of a message is 100 characters. Expanded text in excess of 100 characters is truncated.

Note: CA Endevor SCM keywords can be included in message text. The report prints with the corresponding CA Endevor SCM term. For example:

```
 MSG ('ACTIVITY RECORD FOR CCID: =CCID MISSING');
```

When this message prints on the execution report, =CCID is replaced with the value of the CCID for which the record was not found.

Example

Here is an example of a MSG statement:

```
LABEL DEL0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
   CRITERIA;
      FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
      IFNO;
         RC (8);
         MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
      EIF;
   ECRITERIA;
   POPULATE;
   EPOPULATE;
ELABEL;
```

Criteria Blocks

Use a Criteria block to specify a test that will result in a true/false condition. A test is established by including FIELD statements in this block. If a true condition prevails, the interface populates the InfoMan record with information from other FIELD statements, found in the Populate block.

When no test is specified in the Criteria block, a true condition is assumed to exist, and the Populate block FIELD statements are executed automatically.

Criteria blocks are (Optional) One Criteria block is allowed per Label block.

The syntax is:

```
CRITERIA;
         FIELD statements
         IFYES/IFNO statements
ECRITERIA;
```

The following describes the components of this syntax:

CRITERIA

(Required) CRITERIA must be the first word in a Criteria block, and must be followed by a semicolon.

FIELD statements

FIELD statements establish the conditions for populating InfoMan records. FIELD statements are described in a later section.

IFYES/IFNO statements

Valid only in Criteria block FIELD statements. Used to specify processing when the test established in the FIELD statement is true (IFYES) or false (IFNO). IFYES/IFNO statements are described in a later section.

ECRITERIA

(Required) ECRITERIA must be the last word in a Criteria block, and must be followed by a semicolon. If you want to populate an InfoMan record at this exit point without testing any fields, you must code a null Criteria block. A null Criteria block looks like this:

```
CRITERIA;
ECRITERIA;
```

When coding Criteria blocks, keep in mind that the interface supports storing information only in InfoMan X-type records (freeform text) and R-type records (Response records). Of these two record types, the interface allows you to retrieve for scanning only the R-type records. This means that FIELD statements in Criteria blocks should examine only InfoMan fields that contain R-type records.

Note: For more information about X-type records and R-type records, see your InfoMan documentation.

Example (No Criteria)

Here is an example when there is no criteria, but the user wants to populate an InfoMan record. This use of the CRITERIA statement is common within a Label block associated with a NOTFOUND statement, as shown below.

```
USE (ARCHIVE, BEFORE, STANDALONE);
   NOTFOUND (CREATE,ARC0010);
   FOUND    (ARC0020);
   LABEL ARC0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
      CRITERIA;
      ECRITERIA;
      POPULATE;
         FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
         FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
         FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
            VALUE('API CREATE CHG RECORD');
      EPOPULATE;
   RC(0);
   ELABEL;
```

Example (FIELD statements in both the Criteria and Populate blocks)

Here is an example of FIELD statements in both the Criteria and Populate blocks.

```
LABEL CFM0010;       /* SHIP CONFIRM AFTER - LOG SECTION       */
    CRITERIA;
       FIELD (=PECBSFNM) VALUE ('CONFIRM ');
    ECRITERIA;
    POPULATE;
        /*************************************
        * PREQDEST:  SHIP DESTINATION         *
        * PREQSCNF:  SHIP CONFIRMATIN TYPE    *
        * PREQSRES:  SHIP CONFIRMATION RESULTS *
        * PREQSRCV:  SHIP CONFIRM RC VALUE    *
        *************************************/
        FIELD PANEL(BLG0C010) INDEX(S0E01)
        TEXT ('SHIP:   DESTINATION =PREQDEST CONFIRM TYPE =PREQS  CNF') -
             ('        CONFIRMATION RESULTS  =PREQSRES =PREQSRCV');
    EPOPULATE;
ELABEL;
```

Example (IFYES/IFNO conditions in the Criteria block, in addition to FIELD statements)

Here is an example that includes IFYES/IFNO conditions in the Criteria block, in addition to FIELD statements.

```
LABEL RET0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
    CRITERIA;
        FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
            IFNO;
                RC (8);
                MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
            EIF;
    ECRITERIA;
        POPULATE;
        EPOPULATE;
ELABEL;
```

The Populate Block

Use a Populate block to identify the CA Endevor SCM information to be logged on an InfoMan database record. Populate blocks can only contain FIELD statements.

One Populate block is allowed per Label block, and must follow the Criteria block. If you include a Populate block, you must also include a Criteria block.

Note: A Label block is not required to have the Criteria/Populate blocks.

The syntax is:

```
POPULATE;
     FIELD statements
EPOPULATE;
```

The following describes these components:

POPULATE

(Required) POPULATE must be the first word in a Populate block, and must be followed by a semicolon

FIELD statements

FIELD statements establish the information to be recorded in InfoMan records. FIELD statements are described in a later section.

EPOPULATE

(Required) EPOPULATE must be the last word in a Criteria block, and must be followed by a semicolon. If you do not want to populate an InfoMan record at this exit point, you must code a null Populate block. A null populate block looks like this:

```
POPULATE;
EPOPULATE;
```

Note: The interface supports only InfoMan X-type records (freeformtext) and R-type records (Response records). This means that when coding FIELD statements in a Populate block, you can refer only to InfoMan fields that store X-type or R-type information.

Note: For more information about X-type records and R-type records, see your InfoMan documentation.

Example (null Populate block)

Here is an example of a null Populate block:

```
LABEL RET0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
   CRITERIA;
      FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
         IFNO;
            RC (8);
            MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
         EIF;
   ECRITERIA;
      POPULATE;
      EPOPULATE;
ELABEL;
```

Example (Populate block)

Here is an example of a Populate block:

```
LABEL CFM0010;       /* SHIP CONFIRM AFTER - LOG SECTION       */
   CRITERIA;
      FIELD (=PECBSFNM) VALUE ('CONFIRM ');
   ECRITERIA;
   POPULATE;
       /****************************************
       * PREQDEST:  SHIP DESTINATION            *
       * PREQSCNF:  SHIP CONFIRMATIN TYPE       *
       * PREQSRES:  SHIP CONFIRMATION RESULTS   *
       * PREQSRCV:  SHIP CONFIRM RC VALUE       *
       ****************************************/
       FIELD PANEL(BLG0C010) INDEX(S0E01)
       TEXT ('SHIP: DESTINATION =PREQDEST CONFIRM TYPE =PREQSCNF') -
            ('        CONFIRMATION RESULTS  =PREQSRES =PREQSRCV');
   EPOPULATE;
ELABEL;
```

FIELD Statements

FIELD statements may only appear in Criteria and Populate blocks.

- The interface uses Criteria block FIELD statements to determine a true or false condition. A true condition results in the execution of the Populate Block.
- When FIELD statements appear in the Populate block, the interface uses them to update an InfoMan database record.

Example (FIELD Statement in a Criteria Block, InfoMan Field)

```
LABEL RET0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
    CRITERIA;
        FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
            IFNO;
                RC (8);
                MSG ('INVALID STATUS - MUST BE INITIAL');
            EIF;
    ECRITERIA;
        POPULATE;
        EPOPULATE;
ELABEL;
```

Example (FIELD Statement in a Criteria Block, CA Endevor SCM Field)

```
LABEL CRT0020;       /* LOOKING FOR ACTION: ADD            */
    CRITERIA;
        FIELD (=PACTACTN) VALUE('ADD     ');
    ECRITERIA;
        POPULATE;
        EPOPULATE;
```

Example (FIELD Statement with Equates USE (CREATE, ACTIVITY)

```
LABEL CRT0010;       /* MAINLINE - ALL PKG ACTION POINTS.     */
    CRITERIA;
    ECRITERIA;
       POPULATE;
           FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=PECBUSER);
           FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
           FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
              TEXT('CCID: =PACTCCID IN PACKAGE: =PECBPKID');
           FIELD PANEL(BLG6PTYP) INDEX(S0C09) VALUE('PKG/ACT');
       EPOPULATE;

    CPASS CRT0020;

ELABEL;
```

Example (FIELD Statement with Text)

```
LABEL GRP0020;       /* FOR PACKAGE ACTIONS: GENERATE          */
   RC (0);           /*                      DELETE            */
   CRITERIA;
   ECRITERIA;
       POPULATE;
           /****************************************
           * PACTACTN:  ACTION NAME                *
           * PACTCOMM:  ACTION COMMENT             *
           * PACTSENV:  ENVIRONMENT NAME           *
           * PACTSSYS:  SYSTEM NAME                *
           * PACTSSBS:  SUBSYSTEM NAME             *
           * PACTSTYP:  TYPE                       *
           * PACTSSTG:  STAGE NAME                 *
           * PACTSELM:  ELEMENT NAME               *
           ****************************************/
       FIELD PANEL(BLG0C030) INDEX(S0E01)
       TEXT ('ACTION:  =PACTACTN  =PACTCOMM')
       TEXT ('ELEMENT: =PACTSENV / =PACTSSYS / =PACTSSBS / ') -
           ('=PACTSTYP / =PACTSSTG / =PACTSELM ');
EPOPULATE;
ELABEL;
```

FIELD Statement Syntax

FIELD statement syntax differs slightly depending on whether the syntax is for a Criteria Block or a Populate block. When coding, values within parentheses must remain on the same line.

Criteria FIELD statement syntax to test an InfoMan field is:

```
FIELD PANEL(pname) INDEX(s-word) [LENGTH(length)]
   VALUE({CA Endevor SCM field|'literals'|NONBLANK|BLANK});
   [IFYES statement] [IFNO statement]
```

Note: You cannot specify FIELD statements in a Criteria block associated with a Create Activity Use block.

Criteria FIELD statement syntax to test a CA Endevor SCM field is:

```
FIELD (CA Endevor SCM field) [LENGTH(length)]
   VALUE({'literal'|NONBLANK|BLANK});
   [IFYES statement] [IFNO statement]
```

Populate FIELD statement syntax is:

```
FIELD PANEL(pname) INDEX(s-word) [LENGTH(length)]
    VALUE({CA Endevor SCM field|'literals'|=DATE|=TIME});  or

FIELD PANEL(pname) INDEX(s-word) TEXT(text);
```

Note: Population of CA Endevor SCM field control blocks is not allowed. The syntax components are explained in the next section.

Field Panel (Pname) and Index (S-Word) Clauses

Together, the FIELD PANEL and INDEX clauses identify a particular InfoMan record field.

- When used in a Criteria block FIELD statement, this identifies the field against which the string in the VALUE clause is tested.
- When used in a Populate block FIELD statement, this identifies the field to be populated by the string in the VALUE or TEXT clause.

The Pname identifies the InfoMan panel containing the field to be updated. The S-word identifies the field to be updated. These fields must also be defined in the appropriate PIDT table.

The Length Value

Length is a numeric value that must be greater than zero but less than or equal to 256. The length value determines both the number of characters moved during a MOVE action, and the number of characters compared to determine the true or false condition.

The following table shows how the MOVE action and compare logic use the length value.

If the LENGTH value is	Then the MOVE action or compare logic uses
Greater than the target	The target length
Less than the target	The LENGTH value, with any remaining fields padded with blanks.
Omitted	The smaller of source or target

Note: When CA Endevor SCM fields are converted from binary to character form, all leading zeros are stripped. For example, 'x0010' becomes C'16'. During a compare operation between an InfoMan field and a CA Endevor SCM field that has undergone a binary to character conversion, all leading zeros in the InfoMan field are stripped before the comparison.

The Value Clause - Criteria Block

When used in a Criteria block FIELD statement, the VALUE clause establishes the criteria against which the InfoMan or CA Endevor SCM field is tested. Acceptable test criteria include the following.

CA Endevor SCM field

A value starting with an equal sign identifies a CA Endevor SCM field. For example, VALUE (=PECBUSER). Valid when testing InfoMan fields only.

Literal

Alphanumeric values enclosed in single quotes. For example: VALUE ('UPDATE ')

Blanks

Use this value to test whether the field is empty. A true condition exists when the field contains blanks.

Nonblank

Use this value to test for the presence of information in a field. A true condition exists when the field contains any non-blank characters.

The Value Clause - Populate Block

When used in a Populate block FIELD statement, the VALUE clause identifies the value to be used to populate the InfoMan field. Acceptable populate values include the following.

CA Endevor SCM field

A value starting with an equal sign identifies a CA Endevor SCM field. For example: VALUE (=PECBUSER)

Literal

Alphanumeric values enclosed in single quotes. For example: VALUE ('UPDATE ')

=DATE

Represents the current date. When used, the current date is retrieved to populate an InfoMan database field. Format: mm/dd/yy.

=TIME

Represents the current time. When used, the current time is retrieved to populate an InfoMan database field. Format: hh:mm.

The IFYES/IFNO Statement

IFYES and IFNO clauses may only be used in FIELD statements within the Criteria block.

- The IFYES statement is invoked when the test of the InfoMan or CA Endevor SCM field results in a true condition.
- The IFNO statement is invoked when the test of the InfoMan or CA Endevor SCM field results in a false condition.

IFYES syntax is:

```
IFYES;
   [MSG (text);] [RC (return code);]
EIF; IFNO syntax is:

IFNO;
   [MSG (text);] [RC (return code);]
EIF;
```

TEXT Clauses

Use TEXT clauses to specify information to be used to populate InfoMan fields. You must use this format against InfoMan fields defined as freeform text fields. This format is allowed against other field types.

The text data must be enclosed in single quotes. You can specify text to continue on the same line.

For example:

```
TEXT ('This is an example of a free form text build setup ')  -
     ('statement line one')
```

This text clause is recorded in InfoMan as follows:

```
This is an example of a free form text build setup statement line oe
```

You can also specify text on separate lines. For example:

```
TEXT ('This is an example of a free form text build setup ')
TEXT ('Statement line two')
TEXT ('This is statement line three - Package RC =PECBNVRC')
```

This text clause is recorded in InfoMan as follows:

```
This is an example of a free form text build setup
Statement line two
This is statement line three - Package RC 0
```

You can include a CA Endevor SCM field equate in specified message text:

```
TEXT('CCID: =PACTCCID IN PACKAGE: =PECBPKID');
```

The text recorded in InfoMan for this example includes the CCID for the package action in place of the =PACTCCID equate, and the ID of the associated package in place of the =PECBPKID equate.

CA Endevor SCM Control Block Availability

The E/INFO table uses information stored in exit control blocks when building requests to pass to the InfoMan API. You identify the control block fields you want to use, and the target InfoMan database fields for that information, when you code the Criteria and Populate blocks for your E/INFO table.

Control Blocks for Stand-alone Actions

The information in the $ECBDS and $REQPDS control blocks is available to all stand-alone actions. In addition, the information in the environment ($ENVDS), element ($ELMDS), and file ($FILDS) control blocks is available for either the source (SRC) or the target (TGT) location of an action. The table below summarizes which control blocks are available to which groups of actions.

Actions that can appear together in Use blocks	$ENVDS		$ELMDS		$FILDS		$ECBDS		$REQPDS	
	src	tgt	src	tgt	src	tgt	src	tgt	src	tgt
ADD, UPDATE		x		x	x		x		x	
RETRIEVE, ARCHIVE	x		x			x	x		x	
GENERATE, DELETE	x		x				x		x	
MOVE, TRANSFER, RESTORE	x	x	x	x			x		x	

The sections that follow explain the specific fields from each control block, and the equate values used in the E/INFO table syntax to reference these fields.

Available $ENVDS Fields

The following lists the fields in the $ENVDS control block that are available to the E/INFO interface, the information contained in each field, and the equate value to use in the syntax for the E/INFO table for both source and target locations. The list contains the $ENVDS field, what this contains and the source and target equate values.

This $ENVDS field	Contains this source or target information	And has these equate values
ENVSITE	CA Endevor SCM site ID	■ EV1SITE (source) ■ EV2SITE (target)
ENVENVM	Environment name	■ EV1ENV (source) ■ EV2ENV (target)
ENVSYSTM	System name	■ EV1SYS (source) ■ EV2SYS (target)
ENVSUBSY	Subsystem name	■ EV1SUB (source) ■ EV2SUB (target)

This $ENVDS field	Contains this source or target information	And has these equate values
ENVTYPE	Type name	■ EV1TYP (source) ■ EV2TYP (target)
ENVSTAGE	Stage name	■ EV1STAGE (source) ■ EV2STAGE (target)
ENVSTGID	Stage number for the action (1 or 2)	■ EV1STG# (source) EV2STG# (target)
ENVSTGCD	Stage ID	■ EV1STGID (source) ■ EV2STGID (target)
ENVELEMT	Name of the element	■ EV1ELM (source) ■ EV2ELM (target)
ENVEVER	Version number for the element	■ EV1VER (source) ■ EV2VER (target)
ENVELVL	Level for the element	■ EV1LVL (source) ■ EV2LVL (target)
ENVSPEC	Indicates the kind of location. The location can be: ■ A CA Endevor SCM environment ■ External to CA Endevor SCM	■ EV1LOC (source) ■ EV2LOC (target)
ENVEAOFF	Actual element name length	■ EV1ELMLN (source) ■ EV2ELMLN (target)
ENVEAOFF	Full element name	■ EV1ELMFN (source) ■ EV2ELMFN (target)

Available $ELMDS Fields

The following lists the fields in the $ELMDS control block that are available to the E/INFO interface, the information contained in each field, and the equate value to use in the syntax for the E/INFO table for both source and target locations.

This $ELMDS field	Contains this source or target information	And has these equate values
ELMNAME	First ten characters of the element name	■ EL1NAME (source) ■ EL2NAME (target)
ELMMVERS	Element version	■ EL1VER (source) ■ EL2VER (target)
ELMMLLVL	Element level	■ EL1LVL (source) ■ EL2LVL (target)
ELMM1STL	Base level of the element at this location	■ EL1BASE (source) ■ EL2BASE (target)
ELMMINS	Number of inserts	■ EL1INS (source) ■ EL2INS (target)
ELMMDEL	Number of deletes	■ EL1DEL (source) ■ EL2DEL (target)
ELMCCID	CCID associated with the element	■ EL1CCID (source) ■ EL2CCID (target)
ELMMBTOT	Number of statements in the base level of the element	■ EL1BTOT (source) ■ EL2BTOT (target)
ELMMBDTE	Date associated with the base level	■ EL1BDATE (source) ■ EL2BDATE (target)
ELMMBTIM	Time associated with the base level	■ EL1BTIME (source) ■ EL2BTIME (target)
ELMBCOM	Comment associated with the base level	■ EL1BCOM (source) ■ EL2BCOM (target)
ELMBUSID	User ID associated with the base level	■ EL1BUSER (source) ■ EL2BUSER (target)
ELMNOSRC	Element NoSource indicator flag	■ EL1NOSRC (source) ■ EL2NOSRC (target)

This $ELMDS field	Contains this source or target information	And has these equate values
ELMPUSID	User ID associated with the last generate of the element	■ EL1GUSER (source) ■ EL2GUSER (target)
ELMPDSTP	Date the element was last generated	■ EL1GDATE (source) ■ EL2GDATE (target)
ELMPTSTP	Time the element was last generated	■ EL1GTIME (source) ■ EL2GTIME (target)
ELMPRFLG	Last processor executed against the element: ■ 0--no processor executed ■ 1--processor failed ■ 2--last processor was generate ■ 3--last processor was move ■ 4--last processor was delete ■ 5--restore processor not specified	■ EL1PFG (source) ■ EL2PFG (target)
ELMLPROD	Date of last processor execution	■ EL1PDATE (source) ■ EL2PDATE (target)
ELMLPROT	Time of last processor execution	■ EL1PTIME (source) ■ EL2PTIME (target)
ELMLPROU	User ID associated with last processor executed against the element	■ EL1PUSER (source) ■ EL2PUSER (target)
ELMLPRON	Name of the processor last run against the element	■ EL1PNAME (source) ■ EL2PNAME (target)
ELMMPRC	Processor return code	■ EL1PRC (source) ■ EL2PRC (target)
ELMPCOM	Comment associated with the last action that generated the element	■ EL1PCOM (source) ■ EL2PCOM (target)
ELMMRC	CA Endevor SCM return code (NDVR RC) for the last execution of a processor against the element	■ EL1CRC (source) ■ EL2CRC (target)
ELMFDSN	Name of the data set from which the element was last added, updated, or restored	■ EL1DSN (source) ■ EL2DSN (target)

This $ELMDS field	Contains this source or target information	And has these equate values
ELMFMBR	Member name of the element before it was last added, updated, or restored	■ EL1MBR (source) ■ EL2MBR (target)
ELMMVDTE	Date element was last moved	■ EL1MDATE (source) ■ EL2MDATE (target)
ELMMVTIM	Time element was last moved	■ EL1MTIME (source) ■ EL2MTIME (target)
ELMMUSID	User ID associated with the last move of the element	■ EL1MUSER (source) ■ EL2MUSER (target)
ELMRDSTP	Date element was last retrieved	■ EL1RDATE (source) ■ EL2RDATE (target)
ELMRTSTP	Time element was last retrieved	■ EL1RTIME (source) ■ EL2RTIME (target)
ELMRIID	User ID associated with the last retrieve of the element	■ EL1RUSER (source) ■ EL2RUSER (target)
ELMRCOM	Comment associated with the last retrieve of the element	■ EL1RCOM (source) ■ EL2RCOM (target)
ELMTDSN	Name of the data set to which the element was last retrieved	■ EL1RDSN (source) ■ EL2RDSN (target)
ELMTMBR	Member name of the element at the target data set of the last retrieve	■ EL1RMBR (source) ■ EL2RMBR (target)
ELMCCOM	Current level comment for the element	■ EL1CCOM (source) ■ EL2CCOM (target)
ELMMLDTE	Date associated with the current level of the element	■ EL1LDATE (source) ■ EL1LDATE (target)
ELMMLTME	Time associated with the current level of the element	■ EL1LTIME (source) ■ EL2LTIME (target)
ELMLUSID	User ID associated with the current level of the element	■ EL1LUSER (source) ■ EL2LUSER (target)
ELMMLACT	Most recent action against the element	■ EL1LACT (source) ■ EL2LACT (target)

This $ELMDS field	Contains this source or target information	And has these equate values
ELMMLTOT	Total number of source statements in the current level of the element	■ EL1LTOT (source) ■ EL2LTOT (target)
ELMRCCID	CCID associated with the last RETRIEVE action against the element	■ EL1RCCID (source) ■ EL2RCCID (target)
ELMGCCID	CCID associated with the last GENERATE action against the element	■ EL1GCCID (source) ■ EL2GCCID (target)
ELMODACT	Last action executed against the element that modified the element	■ EL1LMACT (source) ■ EL2LMACT (target)
ELMLCCID	CCID associated with the last action that modified the element	■ EL1LCCID (source) ■ EL2LCCID (target)
ELMLCOMM	Comment associated with the last action that modified the element	■ EL1LCOMM (source) ■ EL2LCOMM (target)
ELMLADT	Date associated with the last action that modified the element	■ EL1MLADT (source) ■ EL2MLADT (target)
ELMLATM	Time associated with the last action that modified the element	■ EL1MLATM (source) ■ EL2MLATM (target)
ELMLUID	User ID associated with the last action that modified the element	■ EL1MLUID (source) ■ EL2MLUID (target)
ELMUPDT	Element name in the delta library	■ EL1NUPDT (source) ■ EL2NUPDT (target)
ELMBASE	Element name in the base library	■ EL1NBASE (source) ■ EL2NBASE (target)
ELMIPCTG	Percentage of inserts deleted	■ EL1IPCTG (source) ■ EL2IPCTG (target)
ELMDPCTG	Percentage of deletes inserted.	■ EL1DPCTG (source) ■ EL2DPCTG (target)
ELMFPESD	Name of footprinted object (ESD) module	■ EL1FPESD (source) ■ EL2FPESD (target)
ELMEDFMT	Delta format for the element	■ EL1EDFMT (source) ■ EL2EDFMT (target)

This $ELMDS field	Contains this source or target information	And has these equate values
ELMLPROV	Version number of last processor used against this element.	■ EL1LPROV (source) ■ EL2LPROV (target)
ELMPROL	Level number of last processor used against this element	■ EL1PROL (source) ■ EL2PROL (target)
ELMFMID	Record format ID	■ EL1FMID (source) ■ EL2FMID (target)
ELMXDNAM	Component list delta member name	■ EL1XDNAM (source) ■ EL2XDNAM (target)
ELMXBVER	Component list delta version number	■ EL1XBVER (source) ■ EL2XBVER (target)
ELMXBTOT	Component list base total	■ EL1XBTOT (source) ■ EL2XBTOT (target)
ELMXLTOT	Component list last level total	■ EL1XLTOT (source) ■ EL2XLTOT (target)
ELMXBLVL	Component list base level number	■ EL1XBLVL (source) ■ EL2XBLVL (target)
ELMXLLVL	Component list last level number	■ EL1XLLVL (source) ■ EL2XLLVL (target)
ELMXINS	Inserts at last level	■ EL1XINS (source) ■ EL2XINS (target)
ELMXDLS	Deletes at last level	■ EL1XDLS (source) ■ EL2XDLS (target)
ELMXRGIN	Component list regression insert percent	■ EL1XRGIN (source) ■ EL2XRGIN (target)
ELMXRGDL	Component list regression delete percent	■ EL1XRGDL (source) ■ EL2XRGDL (target)
ELMXBDTE	Component list base date	■ EL1XBDTE (source) ■ EL2XBDTE (target)
ELMXBTIM	Component list base time	■ EL1XBTIM (source) ■ EL2XBTIM (target)

This $ELMDS field	Contains this source or target information	And has these equate values
ELMXLDTE	Component list last level date	■ EL1XLDTE (source) ■ EL2XLDTE (target)
ELMXLTIM	Component list last level time	■ EL1XLTIM (source) ■ EL2XLTIM (target)
ELMXFLAG	Component list component status flag	■ EL1XFLAG (source) ■ EL2XFLAG (target)
ELMXDFMT	Component list delta format	■ EL1XDFMT (source) ■ EL2XDFMT (target)
ELMXUPDT	Indicates whether component list base is in delta library	■ EL1XUPDT (source) ■ EL2XUPDT (target)
ELMSPKG	ID of the last processed package that included the source for this element	■ EL1SPKG (source) ■ EL2SPKG (target)
ELMSPKGT	Time stamp of the last processed package that included the source for this element	■ EL1SPKGT (source) ■ EL2SPKGT (target)
ELMOPKG	ID of package associated with the outputs of this element	■ EL1OPKG (source) ■ EL2OPKG (target)
ELMOPKGT	Time stamp of package associated with the outputs of this element	■ EL1OPKGT (source) ■ EL2OPKGT (target)
ELMPRGRP	Processor group name	■ EL1PRGRP (source) ■ EL2PRGRP (target)
ELMFFFLG	From file type flag	■ EL1FFFLG (source) ■ EL2FFFLG (target)
ELMTFFLG	To file type flag	■ EL1TFFLG (source) ■ EL2TFFLG (target)
ELMFPAOFF	From path name length	■ EL1FPALN (source) ■ EL2FPALN (target)
ELMFPAOFF	From path name	■ EL1FPANM (source) ■ EL2FPANM (target)
ELMFNAOFF	From file name length	■ EL1FFILN (source) ■ EL2FFILN (target)

This $ELMDS field	Contains this source or target information	And has these equate values
ELMFNAOFF	From file name	■ EL1FFINM (source) ■ EL2FFINM (target)
ELMTPAOFF	To path name length	■ EL1TPALN (source) ■ EL2TPALN (target)
ELMTPAOFF	To path name	■ EL1TPANM (source) ■ EL2TPANM (target)
ELMTNAOFF	To file name length	■ EL1TFILN (source) ■ EL2TFILN (target)
ELMTNAOFF	To file name	■ EL1TFINM (source) ■ EL2TFINM (target)
ELMEAOFF	Element name length	■ EL1ELMLN (source) ■ EL2ELMLN (target)
ELMEAOFF	Full element name	■ EL1ELMFN (source) ■ EL2ELMFN (target)
ELMBVERS	Element base version number	■ EL1BVERS(source) EL2BVERS (target)
ELMXLVER	Component last version level	■ EL1XLVER(source) ■ EL2XLVER(target)

Available $FILDS Fields

Use these equate values for $FILDS fields.

This $FILDS field	Contains this source or target information	And has these equate values
FILDSN	Data set name of the external file used by the current action	■ FI1DSN (source) ■ FI2DSN (target)
FILDSMEM	Member name of the element associated with the current action	■ FI1MBR (source) ■ FI2MBR (target)
FILDDN	DDname associated with the data set named above	■ FI1DDN (source) ■ FI2DDN (target)

This $FILDS field	Contains this source or target information	And has these equate values
FILDSTY	The kind of data set (DA, PSU, PO, IS, and the like)	■ FI1DTY (source) ■ FI2DTY (target)
FILPAOFF	Path name length	■ FI1PALN (source) ■ FI2PALN (target)
FILNAOFF	File name length	■ FI1PANM (source) ■ FI2PANM (target)
FILPAOFF	File name	■ FI1PANM (source) ■ FI2PANM (target)

Available $ECBDS Fields

Use these equate values for $ECBDS fields.

This $ECBDS field	Contains this information	And has this equate value
ECBEXTN	Number identifying the exit being invoked	EXITNO
ECBUSER	User ID associated with the current session. Modifiable by exit 5.	USER
ECBMODE	Indicates whether the current action was requested in foreground (T) or in batch (B).	MODE
ECBFUNC	Action code. For example, 1 is the code for ADD, 7 is the code for MOVE.	ACTID
ECBFUNAM	Literal describing the current action (for example ADD, MOVE).	ACTION

Note: The EXITRC equate value maps to the ECBRTCD field in the $ECBDS control block. This field is updated by the RC keyword value in the E/INFO table syntax.

Available $REQPDS Fields

Use these equate values for $REQPDS fields.

This $REQPDS field	Contains this information	And has this equate value
REQCOMM	Comment associated with the current action.	COMM
REQCCID	CCID associated with the current action.	CCID
REQSISOF	Indicates whether the current action requested a signout override.	SISO
REQSISOC	Applicable for RETRIEVE action. Indicates whether NO SIGNOUT was specified.	COPY
REQEXPND	Applicable for RETRIEVE action. Indicates whether EXPAND INCLUDES was specified.	EXPAND
REQTCOD	Action return code.	ACTRC
REQOVRWR	Applicable for RETRIEVE action. Indicates whether REPLACE was specified.	REPL
REQDEL	Applicable for ADD, ARCHIVE, DELETE, MOVE, TRANSFER, and UPDATE. Indicates whether or not a member or element is deleted after completing the current action.	DEL
REQMOVWH	Applicable for MOVE action. Indicates whether MOVE WITH HISTORY was specified.	WHIST
REQUPDT	Applicable for ADD action. Indicates whether UPDATE IF PRESENT was specified.	UPD
REQORGH	Applicable for the Restore action Retain Generate History option.	RGENHIST
REQOSRCH	Applicable for GENERATE and RETRIEVE actions. Indicates whether CA Endevor SCM is to search the map when building a list.	SEARCH

This $REQPDS field	Contains this information	And has this equate value
REQNOSRC	Applicable for GENERATE actions. Indicates whether CA Endevor SCM is to search the map and use the first found sourced element as input to the generate process. With this option, the found source is not copied back to the target location.	NOSOURCE
REQORSGN	Applicable for MOVE and TRANSFER actions. Indicates whether RETAIN SIGNOUT was specified.	RETSIGN
REQOJUMP	Applicable for MOVE action. Indicates whether CA Endevor SCM allows you to move an element when a version of the element exists at an intermediate stage that is not on the map.	JUMP
REQOSGNI	Applicable for MOVE and TRANSFER actions. Indicates whether or not to sign the target location was specified.	SIGNIN
REQSYNC	Applicable for TRANSFER actions. Indicates whether you want to transfer an element with history when the current level of the element at the target differs from the base level at the from-location.	SYNC

Control Blocks for Packages and Package Actions

The control block information available to package actions is shown below.

Package actions that can appear together in Use blocks	$PECBDS	$PREQPDS	$PHDRDS	$PACTREQ
BACKIN, BACKOUT, CAST, COMMIT, EXECUTE, RESET, REVIEW, SHIP for before-exit and after-exit points	X	X	X	
CREATE/MODIFY				
before	X	X		
after	X	X	X	

Package actions that can appear together in Use blocks	$PECBDS	$PREQPDS	$PHDRDS	$PACTREQ
CREATE ACTIVITY	X	X	X	X
GENPKGID				
before	X	X		
after	X	X		
PDELETE				
before	X	X	X	
after	X	X		

Not all the fields in the $PECBDS control block are available. Those fields that are available are shown in the following list:

This $PECBDS field	Contains this information	And has this equate value
PECBPKID	Package ID	PECBPKID
PECBFNCD	Package function code	PECBFNCD
PECBSFCD	Package subfunction code	PECBFSCD
PECBBACD	Package before/after code	PECBBACD
PECBFNNM	Package function literal	PECBFNNM
PECBSFNM	Package subfunction literal	PECBSFNM
PECBBANM	Package before/after literal	PECBBANM
PECBACTE	Action record existence flag	PECBACTE
PECBAPPE	Approver record existence flag	PECBAPPE
PECBBODE	Backout record existence flag	PECBBODE
PECBNDRC	CA Endevor SCM high return code	PECBNDRC
PECBNDAB	CA Endevor SCM abend code	PECBNDAB
PECBNDMG	CA Endevor SCM message text	PECBNDMG
PECBNDML	CA Endevor SCM message text length	PECBNDML
PECBUSER	TSO user or batch job name	PECBUSER
PECBMODE	Mode setting (online or batch)	PECBMODE

The fields not available include:

- The before/after exit switches (PECBBIBE through PECBSCAF)
- The request for data flags (PECBACTR through PECBSCLR), as well as PECBRQRC, the return code field for user requests for data.
- PECBUECB, PECBUREQ, PECBUFIL, PECBUAPP, PECB#APG.

The following fields are set by the E/INFO interface, through the table syntax:

- PECBRTCD
- PECBMGCD
- PECBMGLN
- PECBMSG

$PREQPDS

Use these equate values for $PREQPDS fields.

This $PREQPDS field	Contains this information	And has this equate value
PREQCOMM	Package comment	PREQCOMM
PREQBOEN	Backout enabled flag	PREQBOEN
PREQDEL	Delete enabled flag	PREQDEL
PREQPSHR	Sharable package flag	PREQPSHR
PREQPAPD	Append package flag	PREQPAPD
PREQEWSD	Execution window start date	PREQEWSD
PREQEWST	Execution window start time	PREQEWST
PREQEWED	Execution window end date	PREQEWED
PREQEWET	Execution window end time	PREQEWET
PREQIPKG	Source package ID (copy subfunction)	PREQIPKG
PREQDEST	Shipment destination	PREQDEST
PREQSTYP	Shipment type	PREQSTYP
PREQSCMP	Ship complementary data set flag	PREQSCMP
PREQSCNF	Ship confirmation type	PREQSCNF
PREQSRES	Ship confirmation result	PREQSRES
PREQSRCV	Ship confirmation return code	PREQSRCV

This $PREQPDS field	Contains this information	And has this equate value
PREQAPGP	Override approver group	PREQAPGP
PREQAPID	Override approver ID	PREQAPID

$PHDRDS

Use these equate values for $PHDRDS fields.

This $PHDRDS field	Contains this information	And has this equate value
PHDRTYPE	Package type (standard or emergency).	PHDRTYPE
PHDRSTAT	Package status	PHDRSTAT
PHDRBOST	Package backout status	PHDRBOST
PHDRCRD	Package creation date	PHDRCRD
PHDRCRT	Package creation time	PHDRCRT
PHDRXRC	Package execution return code	PHDRXRC
PHDRPUD	Date of last update to package SCL	PHDRPUD
PHDRPUT	Time of last update to package SCL	PHDRPUT
PHDRCD	Cast date	PHDRCD
PHDRCT	Cast time	PHDRCT
PHDRAD	Final approval/denial date	PHDRAD
PHDRAT	Final approval/denial time	PHDRAT
PHDRWSD	Execution window start date	PHDRWSD
PHDRWST	Execution window start time	PHDRWST
PHDRWED	Execution window end date	PHDRWED
PHDRWET	Execution window end time	PHDRWET
PHDRXD	Execution date	PHDRXD
PHDRXT	Execution time	PHDRXT
PHDRCMD	Commit date	PHDRCMD
PHDRCMT	Commit time	PHDRCMT
PHDRBOD	Backout date	PHDRBOD
PHDRBOT	Backout time	PHDRBOT

This $PHDRDS field	Contains this information	And has this equate value
PHDRBID	Backin date	PHDRBID
PHDRBIT	Backin time	PHDRBIT
PHDREXD	Date when execution completed	PHDREXD
PHDREXT	Time when execution completed	PHDREXT

$PACTREQ

Use these equate values for $PACTREQ fields.

This $PACTREQ field	Contains this information	And has this equate value
PACTSEQ#	Sequence number of the action.	PACTSEQ#
PACTACTN	CA Endevor SCM action.	PACTACTN
PACTCCID	Action CCID.	PACTCCID
PACTCOMM	Action comment.	PACTCOMM
PACTNDRC	CA Endevor SCM high return code.	PACTNDRC
PACTPRRC	Processor high return code.	PACTPRRC
PACTBEXD	Begin execution date.	PACTBEXD
PACTBEXT	Begin execution time.	PACTBEXT
PACTEEXD	End execution date.	PACTEEXD
PACTEEXT	End execution time.	PACTEEXT
PACTSSIT	Site ID at the source of the action.	PACTSSIT
PACTSENV	Environment at the source location.	PACTSENV
PACTSSYS	System at the source location.	PACTSSYS
PACTSSBS	Subsystem at the source location.	PACTSSBS
PACTSELM	First 10 characters of the element name at the source location.	PACTSELM
PACTSTYP	Type at the source location.	PACTSTYP
PACTSSTG	Stage name at the source location.	PACTSSTG
PACTSSTI	Stage ID at the source location.	PACTSSTI
PACTSVL	Version/level at the source location.	PACTSVL

This $PACTREQ field	Contains this information	And has this equate value
PACTSDD	Delta/base creation date at source location.	PACTSDD
PACTSDT	Delta/base creation time at source location.	PACTSDT
PACTSGD	Date of last GENERATE action at the source location.	PACTSGD
PACTSGT	Time of last GENERATE action at the source location.	PACTSGT
PACTSPD	Date of last processor run at the source location.	PACTSPD
PACTSPT	Time of last processor run at the source location.	PACTSPT
PACTSDSN	External data set name.	PACTSDSN
PACTSMBR	External member name.	PACTSMBR
PACTSPPI	Previous package ID associated with the source.	PACTSPPI
PACTSCTS	Cast time stamp associated with source location.	PACTSCTS
PACTTSIT	Site ID at the target of the action.	PACTTSIT
PACTTENV	Environment at the target location.	PACTTENV
PACTTSYS	System at the target location.	PACTTSYS
PACTTSBS	Subsystem at the target location.	PACTTSBS
PACTTELM	First 10 characters of the element name at the target location.	PACTTELM
PACTTTYP	Type at the target location.	PACTTTYP
PACTTSTG	Stage name at the target location.	PACTTSTG
PACTTSTI	Stage ID at the target location.	PACTTSTI
PACTTDD	Delta/base creation date at target location.	PACTTDD
PACTTDT	Delta/base creation time at target location.	PACTTDT
PACTTGD	Date of last Generate action at the target location.	PACTTGD
PACTTGT	Time of last Generate action at the target location.	PACTTGT

This $PACTREQ field	Contains this information	And has this equate value
PACTTPD	Date of last processor run at the target location.	PACTTPD
PACTTPT	Time of the last processor run at the target location.	PACTTPT
PACTTDSN	External data set name at the target location.	PACTTDSN
PACTTMBR	External member name at the target location.	PACTTMBR
PACTTPPI	Previous package ID associated with the source at the target location.	PACTTPPI
PACTTCTS	Cast time stamp associated with source at the target location.	PACTTCTS
PACTSFLAG	Source file type indicator.	PACTSFLAG
PACTTFLAG	Target file type indicator.	PACTTFLAG
PACTSEAOFF	Source full element name length.	PACTSELMLN
PACTSEAOFF	Source full element name.	PACTSELMFN
PACTSPAOFF	Source path name length.	PACTSPALN
PACTSPAOFF	Source path name.	PACTSPANM
PACTSNAOFF	Source file name length.	PACTSFILN
PACTSNAOFF	Source file name.	PACTSFINM
PACTTEAOFF	Target full element name length.	PACTTELMLN
PACTTEAOFF	Target full element name.	PACTTELMFN
PACTTPAOFF	Target path name length.	PACTTPALN
PACTTPAOFF	Target path name.	PACTTPANM
PACTTNAOFF	Target file name length.	PACTTFILN
PACTTNAOFF	Target file name.	PACTTFINM

Interface Syntax Batch Utility Output Report

When the E/INFO interface utility builds a load module from the table syntax that you submit, it also produces an output report about the compile process. The output report contains the following sections:

- A Utility Syntax Report, which lists the input table syntax.
- An E/INFO Interface Syntax Errors report, listing any messages produced by the utility.

Chapter 4: Setting Up the Interface

This section contains the following topics:

The InfoMan API

This section contains an overview of the InfoMan Application Programming Interface (API), and is intended to provide an orientation for people responsible for implementing the CA Endevor InfoMan Interface.

Note: For more information, consult InfoMan experts at your site, or IBM Tivoli Information Management Application Program Interface Guide SC34-4592-00.

The InfoMan database consists of 16 record types. The interface uses the following four InfoMan record types to store CA Endevor SCM information.

- Problem records, to record information about stand-alone actions. Stand-alone actions are actions that are not associated with a package.
- Change records, to record information about stand-alone actions and packages.
- Change Activity records, to record information about package actions.
- The Add Change Activity record, to relate package actions to a parent package record.

Four actions can be performed against each record: data can be created (Create), data can be updated (Update), inquiries can be made against the record (Inquiry), and the record can be retrieved (Retrieve). The CA Endevor InfoMan Interface supports all these access modes.

InfoMan allows logical views of these records, in the form of Program Interface Data Table (PIDT) tables. PIDT tables define data fields that can be passed between an application (in this case CA Endevor SCM) and the API.

InfoMan comes with a predefined PIDT table for each record type and use. For example, the Problem record type has four PIDT tables: Problem record create, Problem record update, Problem record inquiry, and Problem record retrieve. Each PIDT table provides a view of the problem record.

The InfoMan API uses PIDT tables to access or update data from the InfoMan database.

You can run the interface with the standard PIDT tables, or you can create your own. If you create your own PIDT tables, you must do so for each record type/action that you intend to use, and each custom table must contain certain required fields. These fields are listed later in this chapter.

These PIDT definition statements are fed into the InfoMan BLGUT8 batch utility. This utility produces two machine-loadable data structures for each table definition: the PIDT (Program Interface Data Table) and the Program Interface Pattern Table (PIPT). The PIPT defines the pattern validation criteria for each field found in the PIDT.

How to Set Up the InfoMan Side

Use this checklist when setting up the InfoMan side of the CA Endevor InfoMan Interface.

Note: It may be necessary to obtain detail information from both the CA Endevor SCM and the InfoMan administrators.

1. Decide which CA Endevor SCM information you want to track in InfoMan.
2. Decide which InfoMan record type you want to use to record stand-alone action information: Problem records or Change records.
3. Make sure that all InfoMan fields required by the interface are available.
4. Decide which functions you want to perform against the record types you have selected:

Use this function	To perform the following
Inquiry	Check for the existence of InfoMan records.
Retrieve	Test InfoMan fields for specific values.
Create	Create records in InfoMan.
Update	Update existing records.

5. After making these decisions, build PIDT tables for the record types and functions that you plan to use. For example, if you decide to record CA Endevor SCM package information in InfoMan, you would create PIDT tables for:
   - Inquiry against Change records, for packages.
   - Inquiry against Activity records, for package actions.
   - Retrieve against Change records, for packages.
   - Retrieve against Activity records, for package actions.

- Create Change records, for packages.
- Create Activity records, for package actions.
- Update Change records, for packages.
- Update Activity records, for package actions.

How to Build InfoMan PIDT Tables

The quickest way to get started using the interface is to use the IBM-supplied PIDT tables. If you are using the standard IBM PIDT tables, you can skip this section. Make sure, however, that the InfoMan fields reserved by the interface are not being used at your site to store other information.

If you are not using the standard PIDT tables, first complete the checklist in the previous section. You must then build PIDT tables for the record types (Problem, Change, or Change Activity) that you want to use, and the actions (Inquiry, Create, Update, Retrieve) that you want the interface to perform.

NotFor more information about building PIDT tables, see IBM Tivoli Information Management Application Program Interface Guide SC34-4592-00.

Certain fields must appear in these PIDT tables. These required fields are listed in this section, for each possible record type and use.

NotA four-character (for example IACT) or five-character (for example CPKGA) identifier appears in the following lists. These identifiers are used in the Control block of the batch utility syntax to build the E/INFO table. These identifiers are explained in Chapter 3, "Setting Up the CA Endevor SCM INFO Table."

Inquiry PIDT Required Fields

Inquiry against Change records, for packages (IPKG):

This PIDT field	Is reserved for this information
FIELD PANEL (BLG00000) INDEX(S0B06)	Change type record
FIELD PANEL (BLG6CORQ) INDEX(S0CD4)	Package ID
FIELD PANEL (BLG6PTYP) INDEX(S0C09)	Type: PACKAGE
FIELD PANEL (BLG6ACCN) INDEX(S0CCF)	RNID value

Inquiry against Change Activity records, for package actions (IPKGA):

This PIDT field	Is reserved for this information
FIELD PANEL (BLG0F000) INDEX(S0B07)	Activity type record
FIELD PANEL (BLG6RNOR) INDEX(S0CD0)	Parent RNID
FIELD PANEL (BLG6ACNM) INDEX(S0CBC)	CCID
FIELD PANEL (BLG6PTYP) INDEX(S0C09)	Type: PKG/ACT
FIELD PANEL (BLG6ACCN) INDEX(S0CCF)	RNID value

Inquiry against Change records, for actions (IACT):

This PIDT field	Is reserved for this information
FIELD PANEL (BLG00000) INDEX(S0B06)	Change type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: ACTION
FIELD PANEL(BLG6ACCN) INDEX(S0CCF)	RNID value

Inquiry against Problem records, for actions (IACT):

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0032)	Problem type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: ACTION
FIELD PANEL(BLG6ACCN) INDEX(S0CCF)	RNID value

Retrieve PIDT Required Fields

Retrieve against Change records, for packages (RPKG):

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0B06)	Change type record
FIELD PANEL(BLG6CORQ) INDEX(S0CD4)	Package ID
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: PACKAGE

Retrieve against Change Activity records, for package actions (RPKGA):

This PIDT field	Is reserved for this information
FIELD PANEL(BLGLAL31) INDEX(S0B07)	Activity type record
FIELD PANEL(BLG6RNOR) INDEX(S0CD0)	Parent RNID
FIELD PANEL (BLG6PTYP) INDEX(S0C09)	Type: PKG/ACT
FIELD PANEL(BLG6ACNM) INDEX(S0CBC)	CCID

Retrieve against Change records, for actions (RACT):

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0B06)	Change type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: ACTION

Retrieve against Problem records, for actions (RACT):

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0032)	Problem type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: ACTION

Create PIDT Required Fields

Create Change records for packages (CPKG):

This PIDT field	Is reserved for this information
FIELD PANEL(BLG6ACNM) INDEX(S0CBC)	Package ID
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: PACKAGE
FIELD PANEL(BLG6URN0) INDEX(S0CCF)	RNID value

Create Change Activity records for package actions (CPKGA):

This PIDT field	Is reserved for this information
FIELD PANEL(BLGLAL31) INDEX(S0B07)	Activity type record
FIELD PANEL(BLG6RNOR) INDEX(S0CD0)	Parent RNID

This PIDT field	Is reserved for this information
FIELD PANEL(BLG6CORQ) INDEX(S0CD4)	CCID
FIELD PANEL (BLG6PTYP) INDEX(S0C09)	Type: PKG/ACT
FIELD PANEL(BLG6URN0) INDEX(S0CCF)	RNID value

Create Change records for actions (CACT):

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0B06)	Change type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: package or action
FIELD PANEL(BLG6URN0) INDEX(S0CCF)	RNID value

Create Problem records for actions (CACT):

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0032)	Change type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: package or action
FIELD PANEL(BLG6URN0) INDEX(S0CCF)	RNID value

Update PIDT Required Fields

Update Change records for packages (UPKG):

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0B06)	Change type record

Update Change Activity records for package actions (UPKGA):

This PIDT field	Is reserved for this information
FIELD PANEL(BLGLAL31) INDEX(S0B07)	Activity type record
FIELD PANEL (BLG6PTYP) INDEX(S0C09)	Type: PKG/ACT

Update Change records for actions (UACT):

This PIDT field		Is reserved for this information
FIELD PANEL(BLG00000)	INDEX(S0B06)	Change type record

Update Problem records for actions (UACT):

This PIDT field		Is reserved for this information
FIELD PANEL(BLG00000)	INDEX(S0032)	Problem type record

Chapter 5: Using the Stand-alone Action Table Syntax Template

This section contains the following topics:

The Syntax Template

There are two ways to track stand-alone actions: using InfoMan Change records or using InfoMan Problem records. Although the template described here is designed to track stand-alone actions using InfoMan Change records, you can easily modify the template to include syntax for using Problem records.

The syntax template described in this chapter is contained in member EISCRIP1 in iprfx.iqual.CSIQOPTN. The table built using this syntax updates/references InfoMan Change records. A syntax template for utilizing Problem records is provided in member EISCRIP2 in iprfx.iqual.CSIQOPTN.

If you have not modified standard InfoMan, you can use these syntax templates as-is. Consider, however, reviewing the syntax, editing it as necessary, before submitting it to the Batch Utility.

This sample table syntax uses fields that exist in both Change and Problem record types. If you customize this sample syntax, do not assume that all Change record fields are also defined within Problem records.

The following pages contain a walk-through of EISCRIP1. Even if you want to use InfoMan Problem records (EISCRIP2), studying the walk-through will be beneficial.

IIF Syntax Templates for Stand-Alone Actions

InfoMan Interactive Facility (IIF) users can find syntax templates for change and problem records in iprfx.iqual.CSIQOPTN. Member EISCRXX1 contains the template for building a table to use IIF Change records, and member EISCRXX2 contains a template for building a table to use IIF Problem records.

IIF Sample Syntax

To familiarize you with the syntax, this section provides descriptions for the first few syntax blocks.

```
TABLE NAME (ENDEV02);
```

The TABLE NAME clause starts the table, and names it. In this case the name of the table is ENDEVO2.

Note: For more information, see The Table Statement (see page 33).

```
CTRL;
    IACT (TS0B06I);         /* INQUIRY CHANGE FOR ACTIONS   */
    RACT (TS0B06R);         /* RETRIEVE CHANGE FOR ACTIONS  */
    CACT (TS0B06C);         /* CREATE CHANGE FOR ACTIONS    */
    UACT (TS0B06U);         /* UPDATE CHANGE FOR ACTIONS   */
 ECTRL;
```

This Control block indicates that CA Endevor SCM may request the InfoMan API to:

- Inquire against PIDT table TS0B06I.
- Retrieve PIDT table TS0B06R.
- Create a Change record in PIDT table TS0B06C.
- Update and existing record in PIDT table TS0B06U.

Note: For more information, see The Control Block (see page 34).

```
USE(ADD,BEFORE,STANDALONE);
```

This USE statement indicates that the processing in this Use block applies to the before-exit point for stand-alone ADD actions.

Note: For more information, see Use Blocks (see page 38).

```
NOTFOUND (CREATE,ADD0010);
FOUND    (ADD0020);
```

These NOTFOUND/FOUND statements indicate that:

- If there is no Change record in InfoMan for the CCID associated with an ADD action, the interface is to create a record, using the processing specified in Label block ADD0010.
- If there is a record in InfoMan for the CCID associated with an ADD action, the interface is to execute the processing specified in Label block ADD0020.

Note: For more information, see The FOUND Statement (see page 42).

```
LABEL ADD0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
```

This LABEL statement starts the Label block referred to by the NOTFOUND statement earlier in the syntax.

Note: For more information, see Label Blocks (see page 44).

```
CRITERIA;
ECRITERIA;
```

Even if there are no criteria for populating an InfoMan record, there must be a Criteria block within a Label block, if that block also contains a POPULATE statement. When there are no criteria, code the Criteria block this way.

Note: For more information, see Criteria Blocks (see page 52).

```
POPULATE;
```

This statement starts a Populate block. The FIELD and INDEX statements in a Populate block specify the fields to be populated in InfoMan. The VALUE statements specify the values to use to populate those fields.

Note: For more information, see The Populate Block (see page 55).

```
FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
```

This statement tells the InfoMan API to populate the BLG6REQN field (index S0B59) with the CA Endevor SCM user ID.

Note: For more information, see FIELD Statements (see page 57).

```
FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
```

This statement tells the InfoMan API to populate the BLG6STAT field (index S0BEE) with the literal OPEN.

```
FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
VALUE('API CREATE CHG RECORD');
```

This statement tells the InfoMan API to populate the BLG6DSAB field (index S0E0F) with the text API CREATE CHG RECORD.

Note: For more information, see FIELD Statements (see page 57).

```
EPOPULATE;
```

This statement ends the Populate block.

Note: For more information, see The Populate Block (see page 55).

```
RC(0);
```

This statement tells the API to send a return code of zero back to CA Endevor SCM after successfully executing the processing in this Label block.

Note: For more information, see RC Statements (see page 50).

```
ELABEL;
```

This statement ends Label block ADD0010.

Note: For more information, see Label Blocks (see page 44).

```
LABEL ADD0020;        /* CHANGE RECORD FOUND */
```

This statement starts the label block referenced in the FOUND statement earlier in the syntax.

Note: For more information, see Label Blocks (see page 44).

```
RC (0);
```

This statement tells the API to send a return code of zero back to CA Endevor SCM after successfully executing the processing in this Label block.

Note: For more information, see RC Statements (see page 50).

```
CRITERIA;
```

This is a required statement to start a Criteria block for this Label block.

Note: For more information, see Criteria Blocks (see page 52).

```
FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
```

This statement tells the interface to query field BLG6STAT (index S0BEE) in the InfoMan Change record associated with the CCID for this ADD action. If the value in the field is the literal OPEN, the criteria block is considered true. In this case, this means that it considers the record as found.

Note: For more information, see FIELD Statements (see page 57).

```
ECRITERIA;
```

This is a required statement to end a Criteria block.

Note: For more information, see Criteria Blocks (see page 52).

```
POPULATE;
EPOPULATE;
```

Even if you do not wish to populate an InfoMan record when a Criteria block is true, you must include a Populate block with that Criteria block. This is the way to code a null Populate block.

Note: For more information, see The Populate Block (see page 55).

```
CFAIL ADD0022;
```

This CFAIL statement tells the interface to execute the processing specified in Label block ADD0022 if the Criteria block is false. This would be the case in this example if the InfoMan field BLG6STAT (index S0BEE) did not contain the value OPEN.

Note: For more information, see CPASS/CFAIL Statements (see page 46).

```
ELABEL;
```

This statement ends Label block ADD0020.

Note: For more information, see The ELABEL Statement (see page 45).

```
LABEL ADD0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
```

This statement starts Label block ADD0022.

Note: For more information, see Label Blocks (see page 44).

```
CRITERIA;
  FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
     IFNO;
       RC (8);
       MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
     EIF;
ECRITERIA;
```

This Criteria block tells the interface to query field BLG6STAT (index S0BEE) in the InfoMan Change record associated with the CCID for this ADD action.

- If the value in the field is INITIAL, the criteria block is considered true, and processing continues with the POPULATE statement.
- If the value in the field is not INITIAL, the interface executes the IFNO statement, passing a return code of 8 back to CA Endevor SCM and issuing the message INVALID STATUS - MUST BE INITIAL OR OPEN.

```
POPULATE;
EPOPULATE;
```

Even if you do not wish to populate an InfoMan record when a Criteria block is true, you must include a Populate block with that Criteria block. This is the way to code a null Populate block.

Note: For more information, see The Populate Block (see page 55).

```
ELABEL;
```

This statement ends the ADD0022 Label block.

Note: For more information, see The ELABEL Statement (see page 45).

```
EUSE;
```

This statement ends the Use block for the before-exit point for stand-alone ADD actions.

Note: For more information, see The EUSE Statement (see page 44).

```
USE (ADD, AFTER, STANDALONE);
```

This USE statement starts another Use block. The processing in this Use block is to be invoked for the after-exit point for stand-alone ADD actions.

Note: For more information, see The USE Statement (see page 39).

```
FOUND    (ADD0010);
NOTFOUND (RETURN);
```

These NOTFOUND/FOUND statements indicate that:

- If there is a record in InfoMan for the CCID associated with an ADD action, the interface is to execute the processing specified in Label block ADD0010.
- If there is no Change record in InfoMan for the CCID associated with an ADD action, the interface is to return to CA Endevor SCM.

Note: For more information, see The FOUND Statement (see page 42).

```
LABEL ADD0010;
```

This statement starts Label block ADD0010, referenced in the FOUND statement earlier in the Use block.

Note: For more information, see Label Blocks (see page 44).

```
CRITERIA;
ECRITERIA;
```

Grouping the CRITERIA and ECRITERIA statements this way indicates that there are no criteria to be satisfied after a record is found. This tells the interface to execute the processing specified in the Populate block.

Note: For more information, see Criteria Blocks (see page 52).

```
POPULATE;
  FIELD PANEL(BLG0C010) INDEX(S0E01)
        TEXT ('ACTION PROCESSING COMPLETED RC = =ACTRC')
        TEXT ('ELEMENT: =EV2ENV / =EV2SYS / =EV2SUB ') -

             ('/ =EV2TYP / =EV2STAGE / =EV2ELMFN')

        TEXT ('ACTION: =ACTION USER: =USER DATE: =DATE ') -
             ('TIME: =TIME');
EPOPULATE;
```

This POPULATE statement tells the InfoMan API to populate field BLG0C010 (index S0E01) with a three-line message.

There is one TEXT statement for each line. Lines are continued by placing a hyphen as the last character in the first line. Also note that there is only one semicolon for all three TEXT statements, located at the end of the last one.

The message is built using information from CA Endevor SCM control blocks:

- Line 1 uses the ACTRC field from the $ENVDS block.
- Line 2 uses the EV2ENV, EV2SYS, EV2SUB, EV2TYP, EV2STAGE, and EV2ELMFN fields from the $ENVDS block.
- Line 3 uses the ACTION, USER, DATE, and TIME fields from the $ENVDS block.

```
ELABEL;
```

This statement ends Label block ADD0010.

Note: For more information, see The ELABEL Statement (see page 45).

EUSE; This statement ends the Use block for the after-exit point for stand-alone ADD actions.

Note: For more information, see The EUSE Statement (see page 44).

This concludes the annotated portion of the syntax example. Sample syntax for the remaining stand-alone actions is shown on the following table.

```
USE(ADD, BEFORE, STANDALONE);

NOTFOUND (CREATE,ADD0010);
FOUND    (ADD0020);

LABEL ADD0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
  CRITERIA;
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
             VALUE('API CREATE CHG RECORD');
     EPOPULATE;
  RC(0);
ELABEL;
```

```
LABEL ADD0020;       /* CHANGE RECORD FOUND */
  RC (0);
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
  ECRITERIA;
     POPULATE;
     EPOPULATE;
  CFAIL ADD0022;
ELABEL;

LABEL ADD0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
       IFNO;
         RC (8);
         MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
       EIF;
  ECRITERIA;
     POPULATE;
     EPOPULATE;
ELABEL;
EUSE;

USE (ADD, AFTER, STANDALONE);

FOUND    (ADD0010);
NOTFOUND (RETURN);

LABEL ADD0010;
   CRITERIA;
   ECRITERIA;
        POPULATE;
          FIELD PANEL(BLG0C010) INDEX(S0E01)
                TEXT ('ACTION PROCESSING COMPLETED RC = =ACTRC')
                TEXT ('ELEMENT: =EV2ENV / =EV2SYS / =EV2SUB ') -
                     ('/ =EV2TYP / =EV2STAGE / =EV2ELMFN')
                TEXT ('ACTION: =ACTION USER: =USER DATE: =DATE ') -
                     ('TIME: =TIME');
        EPOPULATE;
ELABEL;
EUSE;

USE (UPDATE, BEFORE, STANDALONE);

NOTFOUND (CREATE,UPT0010);
FOUND    (UPT0020);

LABEL UPT0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
```

```
  CRITERIA;
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
             VALUE('API CREATE CHG RECORD');
     EPOPULATE;
  RC(0);
ELABEL;

LABEL UPT0020;       /* CHANGE RECORD FOUND */
  RC (0);
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
  ECRITERIA;
     POPULATE;
     EPOPULATE;
  CFAIL UPT0022;
ELABEL;

LABEL UPT0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
       IFNO;
         RC (8);
         MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
       EIF;
  ECRITERIA;
     POPULATE;
     EPOPULATE;
ELABEL;

EUSE;

USE (UPDATE, AFTER, STANDALONE);

FOUND    (UPT0010);
NOTFOUND (RETURN);

LABEL UPT0010;
   CRITERIA;
   ECRITERIA;
        POPULATE;
          FIELD PANEL(BLG0C010) INDEX(S0E01)
                TEXT ('ACTION PROCESSING COMPLETED RC = =ACTRC')
                TEXT ('ELEMENT: =EV2ENV / =EV2SYS / =EV2SUB ') -
                     ('/ =EV2TYP / =EV2STAGE / =EV2ELMFN')
                TEXT ('ACTION: =ACTION USER: =USER DATE: =DATE ') -
```

```
                    ('TIME: =TIME');
       EPOPULATE;
ELABEL;
EUSE;

USE (RESTORE,BEFORE, STANDALONE);

NOTFOUND (CREATE,RST0010);
FOUND    (RST0020);

LABEL RST0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
  CRITERIA;
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
             VALUE('API CREATE CHG RECORD');
     EPOPULATE;
  RC(0);
ELABEL;

LABEL RST0020;       /* CHANGE RECORD FOUND */
  RC (0);
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
  ECRITERIA;
     POPULATE;
     EPOPULATE;
  CFAIL RST0022;
ELABEL;

LABEL RST0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
       IFNO;
         RC (8);
         MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
       EIF;
  ECRITERIA;
     POPULATE;
     EPOPULATE;
ELABEL;

EUSE;

USE (RESTORE,AFTER, STANDALONE);

FOUND    (RST0010);
```

```
NOTFOUND (RETURN);

LABEL RST0010;
   CRITERIA;
   ECRITERIA;
        POPULATE;
          FIELD PANEL(BLG0C010) INDEX(S0E01)
            TEXT ('ACTION PROCESSING COMPLETED RC = =ACTRC')
            TEXT ('SOURCE ELEMENT: =EV1ENV / =EV1SYS / =EV1SUB ') -
                 ('/ =EV1TYP / =EV1STG# / =EV1ELMFN')
            TEXT ('TARGET ELEMENT: =EV2ENV / =EV2SYS / =EV2SUB ') -
                 ('/ =EV2TYP / =EV2STAGE / =EV2ELMFN')
            TEXT ('ACTION: =ACTION USER: =USER DATE: =DATE ') -
                 ('TIME: =TIME');
        EPOPULATE;
ELABEL;
EUSE;

USE (GENERATE, BEFORE, STANDALONE);

NOTFOUND (CREATE,GEN0010);
FOUND    (GEN0020);

LABEL GEN0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
  CRITERIA;
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
             VALUE('API CREATE CHG RECORD');
     EPOPULATE;
  RC(0);
ELABEL;

LABEL GEN0020;       /* CHANGE RECORD FOUND */
  RC (0);
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
  ECRITERIA;
     POPULATE;
     EPOPULATE;
  CFAIL GEN0022;
ELABEL;

LABEL GEN0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
       IFNO;
```

```
        RC (8);
        MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
      EIF;
  ECRITERIA;
     POPULATE;
     EPOPULATE;
ELABEL;

EUSE;

USE (GENERATE ,AFTER, STANDALONE);

FOUND    (GEN0010);
NOTFOUND (RETURN);

LABEL GEN0010;
   CRITERIA;
   ECRITERIA;
        POPULATE;
          FIELD PANEL(BLG0C010) INDEX(S0E01)
                TEXT ('ACTION PROCESSING COMPLETED RC = =ACTRC')
                TEXT ('ELEMENT: =EV1ENV / =EV1SYS / =EV1SUB ') -
                     ('/ =EV1TYP / =EV1STAGE / =EV1ELMFN')
                TEXT ('ACTION: =ACTION USER: =USER DATE: =DATE ') -
                     ('TIME: =TIME');
        EPOPULATE;
ELABEL;
EUSE;

USE (RETRIEVE, BEFORE, STANDALONE);

NOTFOUND (CREATE,RET0010);
FOUND    (RET0020);

LABEL RET0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
  CRITERIA;
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
             VALUE('API CREATE CHG RECORD');
     EPOPULATE;
  RC(0);
ELABEL;

LABEL RET0020;       /* CHANGE RECORD FOUND */
  RC (0);
  CRITERIA;
```

```
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
  ECRITERIA;
     POPULATE;
     EPOPULATE;
  CFAIL RET0022;
ELABEL;

LABEL RET0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
       IFNO;
         RC (8);
         MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
       EIF;
  ECRITERIA;
     POPULATE;
     EPOPULATE;
ELABEL;


EUSE;

USE (RETRIEVE, AFTER, STANDALONE);

FOUND    (RET0010);
NOTFOUND (RETURN);

LABEL RET0010;
   CRITERIA;
   ECRITERIA;
        POPULATE;
          FIELD PANEL(BLG0C010) INDEX(S0E01)
                TEXT ('ACTION PROCESSING COMPLETED RC = =ACTRC')
                TEXT ('ELEMENT: =EV1ENV / =EV1SYS / =EV1SUB ') -
                     ('/ =EV1TYP / =EV1STAGE / =EV1ELMFN')
                TEXT ('ACTION: =ACTION USER: =USER DATE: =DATE ') -
                     ('TIME: =TIME');
        EPOPULATE;
ELABEL;
EUSE;


USE (ARCHIVE, BEFORE, STANDALONE);

NOTFOUND (CREATE,ARC0010);
FOUND    (ARC0020);

LABEL ARC0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
  CRITERIA;
```

```
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
             VALUE('API CREATE CHG RECORD');
     EPOPULATE;
  RC(0);
ELABEL;

LABEL ARC0020;       /* CHANGE RECORD FOUND */
  RC (0);
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
  ECRITERIA;
     POPULATE;
     EPOPULATE;
  CFAIL ARC0022;
ELABEL;

LABEL ARC0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
       IFNO;
         RC (8);
         MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
       EIF;
  ECRITERIA;
     POPULATE;
     EPOPULATE;
ELABEL;


EUSE;

USE (ARCHIVE, AFTER, STANDALONE);

FOUND    (ARC0010);
NOTFOUND (RETURN);

LABEL ARC0010;
   CRITERIA;
   ECRITERIA;
        POPULATE;
          FIELD PANEL(BLG0C010) INDEX(S0E01)
                TEXT ('ACTION PROCESSING COMPLETED RC = =ACTRC')
                TEXT ('ELEMENT: =EV1ENV / =EV1SYS / =EV1SUB ') -
                     ('/ =EV1TYP / =EV1STAGE / =EV1ELMFN')
                TEXT ('TARGET: =FI2DSN =FI2MBR')
```

```
                TEXT ('ACTION: =ACTION USER: =USER DATE: =DATE ') -
                     ('TIME: =TIME');
        EPOPULATE;
ELABEL;
EUSE;


USE (DELETE, BEFORE, STANDALONE);

NOTFOUND (CREATE,DEL0010);
FOUND    (DEL0020);

LABEL DEL0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
  CRITERIA;
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
             VALUE('API CREATE CHG RECORD');
     EPOPULATE;
  RC(0);
ELABEL;

LABEL DEL0020;       /* CHANGE RECORD FOUND */
  RC (0);
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
  ECRITERIA;
     POPULATE;
     EPOPULATE;
  CFAIL DEL0022;
ELABEL;

LABEL DEL0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
       IFNO;
         RC (8);
         MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
       EIF;
  ECRITERIA;
     POPULATE;
     EPOPULATE;
ELABEL;

EUSE;

USE (DELETE, AFTER, STANDALONE);
```

```
FOUND    (DEL0010);
NOTFOUND (RETURN);

LABEL DEL0010;
   CRITERIA;
   ECRITERIA;
        POPULATE;
          FIELD PANEL(BLG0C010) INDEX(S0E01)
                TEXT ('ACTION PROCESSING COMPLETED RC = =ACTRC')
                TEXT ('ELEMENT: =EV1ENV / =EV1SYS / =EV1SUB ') -
                     ('/ =EV1TYP / =EV1STAGE / =EV1ELMFN')
                TEXT ('ACTION: =ACTION USER: =USER DATE: =DATE ') -
                     ('TIME: =TIME');
        EPOPULATE;
ELABEL;
EUSE;

USE (MOVE, BEFORE, STANDALONE);

NOTFOUND (CREATE,MOV0010);
FOUND    (MOV0020);

LABEL MOV0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
  CRITERIA;
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
             VALUE('API CREATE CHG RECORD');
     EPOPULATE;
  RC(0);
ELABEL;

LABEL MOV0020;       /* CHANGE RECORD FOUND */
  RC (0);
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
  ECRITERIA;
     POPULATE;
     EPOPULATE;
  CFAIL MOV0022;
ELABEL;

LABEL MOV0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
       IFNO;
```

```
        RC (8);
        MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
      EIF;
  ECRITERIA;
     POPULATE;
     EPOPULATE;
ELABEL;

EUSE;

USE (MOVE, AFTER, STANDALONE);

FOUND    (MOV0010);
NOTFOUND (RETURN);

LABEL MOV0010;
   CRITERIA;
   ECRITERIA;
        POPULATE;
          FIELD PANEL(BLG0C010) INDEX(S0E01)
                TEXT ('ACTION PROCESSING COMPLETED RC = =ACTRC')
                TEXT ('SOURCE ELEMENT: =EV1ENV / =EV1SYS / =EV1SUB ') -
                     ('/ =EV1TYP / =EV1STAGE / =EV1ELMFN')
                TEXT ('TARGET ELEMENT: =EV2ENV / =EV2SYS / =EV2SUB ') -
                     ('/ =EV2TYP / =EV2STAGE / =EV2ELMFN')
                TEXT ('ACTION: =ACTION USER: =USER DATE: =DATE ') -
                     ('TIME: =TIME');
        EPOPULATE;
ELABEL;
EUSE;

USE (TRANSFER, BEFORE, STANDALONE);

NOTFOUND (CREATE,TRA0010);
FOUND    (TRA0020);

LABEL TRA0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
  CRITERIA;
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=USER);
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
             VALUE('API CREATE CHG RECORD');
     EPOPULATE;
  RC(0);
ELABEL;

LABEL TRA0020;       /* CHANGE RECORD FOUND */
```

```
  RC (0);
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
  ECRITERIA;
     POPULATE;
     EPOPULATE;
  CFAIL TRA0022;
ELABEL;

LABEL TRA0022;       /* STATUS NOT OPEN - CHECK FOR INITIAL  */
  CRITERIA;
    FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('INITIAL');
       IFNO;
         RC (8);
         MSG ('INVALID STATUS - MUST BE INITIAL OR OPEN');
       EIF;
  ECRITERIA;
     POPULATE;
     EPOPULATE;
ELABEL;

EUSE;

USE (TRANSFER, AFTER, STANDALONE);

FOUND    (TRA0010);
NOTFOUND (RETURN);

LABEL TRA0010;
   CRITERIA;
   ECRITERIA;
        POPULATE;
          FIELD PANEL(BLG0C010) INDEX(S0E01)
                TEXT ('ACTION PROCESSING COMPLETED RC = =ACTRC')
                TEXT ('SOURCE ELEMENT: =EV1ENV / =EV1SYS / =EV1SUB ') -
                     ('/ =EV1TYP / =EV1STAGE / =EV1ELMFN')
                TEXT ('TARGET ELEMENT: =EV2ENV / =EV2SYS / =EV2SUB ') -
                     ('/ =EV2TYP / =EV2STAGE / =EV2ELMFN')
                TEXT ('ACTION: =ACTION USER: =USER DATE: =DATE ') -
                     ('TIME: =TIME');
        EPOPULATE;
ELABEL;
EUSE;



ETABLE;
```

Chapter 6: Building the CA Endevor SCM Info Table Using Table Syntax

This section contains the following topics:

Package and Package Action Sample Syntax

This section contains sample table syntax for packages and package actions. The sample is for your use when building the E/INFO table for your site. The sample described is located in member EISCRIP3 in iprfx.iqual.CSIQOPTN. Several lines are annotated at the beginning of the sample, for your reference.

InfoMan Interactive Facility (IIF) users can find a syntax template for packages and package actions in member EISCRXX3 in iprfx.iqual.CSIQOPTN.

```
TABLE NAME (ENDEV03);
```

This statement starts the table. The name of the table is ENDEV03. There must be an ETABLE statement at the end of the table.

Note: For more information, see The Label Statement (see page 45).

```
CTRL;
  IPKG (TS0B06I);         /* INQUIRY CHANGE TABLE FOR PACKAGES       */
  IPKGA(TS0B07I);         /* INQUIRY ACTIVITY  TABLE FOR PKG ACTIONS */
  RPKG (TS0B06R);         /* RETRIEVE CHANGE TABLE FOR PACKAGES      */
  RPKGA(TS0B07R);         /* RETRIEVE ACTIVITY TABLE FOR PKG ACTIONS */
  CPKG (TS0B06C);         /* CREATE CHANGE TABLE FOR PACKAGES        */
  CPKGA(TS0B07C);         /* CREATE ACTIVITY TABLE FOR PKG ACTIONS   */
  UPKG (TS0B06U);         /* UPDATE CHANGE TABLE FOR PACKAGES        */
  UPKGA(TS0B07U);         /* UPDATE ACTIVITY TABLE FOR PKG ACTIONS   */
  AACT (TS0B06A);         /* ADD CHANGE ACTIVITY TABLE FOR PKG ACTION*/
ECTRL;
```

This Control block indicates that CA Endevor SCM may request the InfoMan API to:

- Inquire about package (IPKG) and package action (IPKGA) records.
- Retrieve package (RPKG) and package action (RPKGA) records.

- Create a package (CPKG) and package action (CPKGA) records.
- Update package (UPKG) and package action (UPKGA) records.
- Add change activity (AACT) records for package actions.

Note: For more information, see The Control Block (see page 34).

```
/*********************************************
 * EXIT POINTS: CREATE/AFTER   MODIFY/ AFTER    *
 *********************************************/
USE ((CREATE, AFTER),
     (MODIFY, AFTER));
```

This USE statement indicates that the processing in this Use block applies to the after-exit point for the CREATE PACKAGE and MODIFY PACKAGE actions.

Note: For more information, see Use Blocks (see page 38).

```
NOTFOUND (CREATE,CMD0010);
FOUND    (CMD0020);
```

These NOTFOUND/FOUND statements indicate that

- If there is no Change record in InfoMan for a package, the interface is to create a record, using the processing specified in Label block CMD0010.
- If there is a record in InfoMan for the package, the interface is to execute the processing specified in Label block CMD0020.

Note: For more information, see The FOUND Statement (see page 42).

```
LABEL CMD0010;       /* CHANGE RECORD NOT FOUND: CREATE ONE */
```

This LABEL statement starts the Label block referred to by the NOTFOUND statement earlier in the syntax.

Note: For more information, see Label Blocks (see page 44).

```
CRITERIA;
ECRITERIA;
```

Even if there are no criteria for populating an InfoMan record, there must be a Criteria block within a Label block, if that block also contains a POPULATE statement. When there are no criteria, code the Criteria block this way.

Note: For more information, see Criteria Blocks (see page 52).

```
POPULATE;
     /*************************************
      * PECBUSER:  CURRENT USER ID         *
      *************************************/
  FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=PECBUSER);
  FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
  FIELD PANEL(BLG0C010) INDEX(S0E01)
        VALUE('E/INFO CREATED CHANGE RECORD');
EPOPULATE;
```

This POPULATE block tells the InfoMan API what information to put in the Change record that it creates. In this example the API populates three fields:

- The BLG6REQN field (index S0B59) with the value in the PECBUSER field of the CA Endevor SCM package exit control block ($PECBDS).
- The BLG6STAT field (index S0BEE) with the value OPEN.
- The BLG0C010 field (index S0E01) with the value E/INFO CREATED CHANGE RECORD.

Note: For more information, see Criteria Blocks (see page 52).

```
CPASS CMD0020;
```

The CPASS statement tells the interface to execute the processing specified in Label block CMD0020 if the API populates the new record successfully.

Note: For more information, see CPASS/CFAIL Statements (see page 46).

```
ELABEL;
```

This statement ends Label block CMD0010.

Note: For more information, see The ELABEL Statement (see page 45).

```
LABEL CMD0020;
```

This statement starts the label block referenced in the CPASS statement in the preceding Label block.

Note: For more information, see Label Blocks (see page 44).

```
RC (0);
```

This statement tells the API to send a return code of zero back to CA Endevor SCM after successfully executing the processing in this Label block.

Note: For more information, see RC Statements (see page 50).

```
CRITERIA;
ECRITERIA;
```

Grouping the CRITERIA and ECRITERIA statements this way indicates that there are no criteria to be satisfied. This tells the interface to execute the processing specified in the Populate block.

Note: For more information, see Criteria Blocks (see page 52).

```
POPULATE;
        /*****************************************
         * PREQCOMM:  PACKAGE COMMENT            *
         * PREQEWSD:  EXECUTION WINDOW START DATE *
         * PREQEWST:  EXECUTION WINDOW START TIME *
         * PREQEWED:  EXECUTION WINDOW END DATE   *
         * PREQEWET:  EXECUTION WINDOW END TIME   *
         * PECBUSER:  CURRENT USER ID             *
         * PECBFNNM:  PACKAGE FUNCTION            *
         * PECBBANM:  BEFORE OR AFTER             *
         * PECBNDRC:  NDVR HIGH RETURN CODE       *
         * PHDRSTAT:  PACKAGE STATUS              *
         * PHDRCRD:   PACKAGE CREATION DATE       *
         * PHDRCRT:   PACKAGE CREATION TIME       *
         *****************************************/
```

This POPULATE statement is followed by a multi-line comment expanding the names of the CA Endevor SCM control block fields used to populate InfoMan fields in the following FIELD and TEXT statements.

Note: For more information, see The Populate Block (see page 55).

```
FIELD PANEL(BLG6DSAB) INDEX(S0E0F) VALUE(=PREQCOMM);
FIELD PANEL(BLG6SCHD) INDEX(S0C41) VALUE(=PREQEWSD);
FIELD PANEL(BLG6SCHT) INDEX(S0C6E) VALUE(=PREQEWST);
FIELD PANEL(BLG6TARD) INDEX(S0C42) VALUE(=PREQEWED);
FIELD PANEL(BLG6TART) INDEX(S0C6F) VALUE(=PREQEWET);
FIELD PANEL(BLG600CN) INDEX(S0B5C) VALUE(=PECBUSER);
FIELD PANEL(BLG6PHAC) INDEX(S0C0B) VALUE(=PECBFNNM);
```

These FIELD statements specify the InfoMan fields that are to contain CA Endevor SCM information, and the values that the API is to use to populate these fields. The comment earlier in the Populate block identifies the information referred to by the equate value.

Note: For more information, see FIELD Statements (see page 57).

```
FIELD PANEL(BLG0C010) INDEX(S0E01)
  TEXT ('PACKAGE PROCESSING COMPLETED RC = =PECBNDRC')
  TEXT ('INFO ACCESSED FROM PACKAGE EXIT =PECBBANM / =PECBFNNM')
  TEXT ('PKG STATUS:  =PHDRSTAT  USER: =PECBUSER')
  TEXT ('CREATE DATE: =PHDRCRD   TIME: =PHDRCRT');
```

These TEXT statements tell the interface to populate field BLGC010 (index S0E01) with a four-line message.

There is one TEXT statement for each line. Lines are continued by placing a hyphen as the last character in the first line. Also note that there is only one semicolon for all three TEXT statements, located at the end of the last statement.

The message is built using information from CA Endevor SCM control blocks.

- Line 1 uses the PECBNDRC field from the $PECBDS block.
- Line 2 uses the PECBBANM and PECBFNNM fields from the $PECBDS block.
- Line 3 uses the PHDRSTAT field from the PHDRDS block and the PECBUSER field from the $PECBDS block.
- Line 4 uses the PHDRCRD and PHDRCRT fields from the PHDRDS block.

```
EPOPULATE;
```

This statement ends the Populate block.

Note: For more information, see The Populate Block (see page 55).

```
ELABEL;
```

This statement ends Label block CMD0020.

Note: For more information, see The ELABEL Statement (see page 45).

```
EUSE;
```

This statement ends the Use block for the after-exit point for the CREATE PACKAGE and MODIFY PACKAGE actions.

Note: For more information, see The EUSE Statement (see page 44).

```
USE (CAST, AFTER);

FOUND    (CST0020);
NOTFOUND (CREATE, CST0010);

LABEL CST0010;       /* CHANGE RECORD NOT FOUND:  CREATE ONE. */
   CRITERIA;
   ECRITERIA;
     POPULATE;
                     /*************************************
                      * PECBUSER:  CURRENT USER ID         *
                      *************************************/
       FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=PECBUSER);
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
     EPOPULATE;
   CPASS CST0020;
ELABEL;

LABEL CST0020;
  RC (0);
  CRITERIA;
  ECRITERIA;
     POPULATE;
                     /*****************************************
                      * PREQCOMM:  PACKAGE COMMENT            *
                      * PREQEWSD:  EXECUTION WINDOW START DATE *
                      * PREQEWST:  EXECUTION WINDOW START TIME *
                      * PREQEWED:  EXECUTION WINDOW END DATE   *
                      * PREQEWET:  EXECUTION WINDOW END TIME   *
                      * PECBUSER:  CURRENT USER ID             *
                      * PECBFNNM:  PACKAGE FUNCTION            *
                      * PECBBANM:  BEFORE OR AFTER             *
                      * PECBNDRC:  NDVR HIGH RETURN CODE       *
                      * PHDRSTAT:  PACKAGE STATUS              *
                      * PHDRCRD:   PACKAGE CREATION DATE       *
                      * PHDRCRT:   PACKAGE CREATION TIME       *
                      * PHDRCD:    PACKAGE CAST DATE           *
                      * PHDRCT:    PACKAGE CAST TIME           *
                      *****************************************/
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F) VALUE(=PREQCOMM);
       FIELD PANEL(BLG6SCHD) INDEX(S0C41) VALUE(=PREQEWSD);
       FIELD PANEL(BLG6SCHT) INDEX(S0C6E) VALUE(=PREQEWST);
       FIELD PANEL(BLG6TARD) INDEX(S0C42) VALUE(=PREQEWED);
       FIELD PANEL(BLG6TART) INDEX(S0C6F) VALUE(=PREQEWET);
       FIELD PANEL(BLG600CN) INDEX(S0B5C) VALUE(=PECBUSER);
       FIELD PANEL(BLG6PHAC) INDEX(S0C0B) VALUE(=PECBFNNM);
       FIELD PANEL(BLG0C010) INDEX(S0E01)
        TEXT ('PACKAGE PROCESSING COMPLETED RC = =PECBNDRC')
        TEXT ('INFO ACCESSED FROM PACKAGE EXIT =PECBBANM / =PECBFNNM')
```

```
        TEXT ('PKG STATUS:  =PHDRSTAT  USER: =PECBUSER')
        TEXT ('CREATE DATE: =PHDRCRD   TIME: =PHDRCRT')
        TEXT ('CAST   DATE: =PHDRCD    TIME: =PHDRCT');
     EPOPULATE;
ELABEL;

EUSE;


/*********************************************
 * EXIT POINT:  CREATE/ACTIVITY. INVOKED AFTER  *
 * THE CAST AFTER EXIT. THIS INFO EXIT WILL     *
 * CREATE AN ACTIVITY RECORD PER PACKAGE ACTION *
 * BY CCID.                                    *
 *********************************************/

USE (CREATE, ACTIVITY);

LABEL CRT0010;       /* MAINLINE - ALL PKG ACTION POINTS.     */
   CRITERIA;
   ECRITERIA;
     POPULATE;
                     /*************************************
                      * PECBUSER:  CURRENT USER ID          *
                      *************************************/
       FIELD PANEL(BLG6REQN) INDEX(S0B59) VALUE(=PECBUSER);
       FIELD PANEL(BLG6STAT) INDEX(S0BEE) VALUE('OPEN');
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F)
             TEXT ('CCID: =PACTCCID IN PACKAGE: =PECBPKID');
     EPOPULATE;
   CPASS CRT0020;
ELABEL;

LABEL CRT0020;       /* LOOKING FOR ACTION: ADD             */
  CRITERIA;
    FIELD (=PACTACTN) VALUE('ADD     ');
  ECRITERIA;
     POPULATE;
     EPOPULATE;

  CPASS GRP0010;
  CFAIL CRT0025;
ELABEL;

LABEL CRT0025;       /* LOOKING FOR ACTION: UPDATE          */
  CRITERIA;
    FIELD (=PACTACTN) VALUE('UPDATE  ');
  ECRITERIA;
     POPULATE;
```

```
     EPOPULATE;

  CPASS GRP0010;
  CFAIL CRT0030;
ELABEL;

LABEL CRT0030;       /* LOOKING FOR ACTION: RESTORE         */
  CRITERIA;
    FIELD (=PACTACTN) VALUE('RESTORE ');
  ECRITERIA;
     POPULATE;
     EPOPULATE;

  CPASS GRP0010;
  CFAIL CRT0035;
ELABEL;

LABEL CRT0035;       /* LOOKING FOR ACTION: GENERATE        */
  CRITERIA;
    FIELD (=PACTACTN) VALUE('GENERATE');
  ECRITERIA;
     POPULATE;
     EPOPULATE;

  CPASS GRP0020;
  CFAIL CRT0040;
ELABEL;

LABEL CRT0040;       /* LOOKING FOR ACTION: DELETE          */
  CRITERIA;
    FIELD (=PACTACTN) VALUE('DELETE  ');
  ECRITERIA;
     POPULATE;
     EPOPULATE;

  CPASS GRP0020;
  CFAIL CRT0045;
ELABEL;

LABEL CRT0045;       /* LOOKING FOR ACTION: RETRIEVE        */
  CRITERIA;
    FIELD (=PACTACTN) VALUE('RETRIEVE');
  ECRITERIA;
     POPULATE;
     EPOPULATE;

  CPASS GRP0030;
  CFAIL CRT0050;
ELABEL;
```

```
LABEL CRT0050;       /* LOOKING FOR ACTION: ARCHIVE         */
  CRITERIA;
    FIELD (=PACTACTN) VALUE('ARCHIVE ');
  ECRITERIA;
     POPULATE;
     EPOPULATE;

  CPASS GRP0020;
  CFAIL CRT0055;
ELABEL;

LABEL CRT0055;       /* LOOKING FOR ACTION: MOVE            */
  CRITERIA;
    FIELD (=PACTACTN) VALUE('MOVE    ');
  ECRITERIA;
     POPULATE;
     EPOPULATE;

  CPASS GRP0040;
  CFAIL CRT0060;
ELABEL;

LABEL CRT0060;       /* LOOKING FOR ACTION: TRANSFER        */
  CRITERIA;
    FIELD (=PACTACTN) VALUE('TRANSFER');
  ECRITERIA;
     POPULATE;
     EPOPULATE;

  CPASS GRP0040;
ELABEL;

LABEL GRP0010;       /* FOR PACKAGE ACTION: ADD             */
  RC (0);            /*                     UPDATE          */
  CRITERIA;          /*                     RESTORE         */
  ECRITERIA;
     POPULATE;
                     /*****************************************
                      * PACTACTN:  ACTION NAME                *
                      * PACTCOMM:  ACTION COMMENT             *
                      * PACTSDSN:  DATA SET NAME              *
                      * PACTSMBR:  MEMBER                     *
                      * PACTTENV:  ENVIRONMENT NAME           *
                      * PACTTSYS:  SYSTEM NAME                *
                      * PACTTSBS:  SUBSYSTEM NAME             *
                      * PACTTTYP:  TYPE                       *
                      * PACTTSTG:  STAGE NAME                 *
                      * PACTTELM:  ELEMENT NAME               *
```

```
                    ****************************************/
      FIELD PANEL(BLG0C030) INDEX(S0E01)
       TEXT ('ACTION:  =PACTACTN  =PACTCOMM')
       TEXT ('ELEMENT: =PACTTENV / =PACTTSYS / =PACTTSBS / ') -
            ('=PACTTTYP / =PACTTSTG / =PACTTELMFN ');
    EPOPULATE;
  CPASS GRP0050;
ELABEL;


LABEL GRP0020;       /* FOR PACKAGE ACTIONS: GENERATE          */
  RC (0);            /*                      DELETE            */
  CRITERIA;          /*                      ARCHIVE           */
  ECRITERIA;
    POPULATE;
                    /****************************************
                     * PACTACTN:  ACTION NAME                *
                     * PACTCOMM:  ACTION COMMENT             *
                     * PACTSENV:  ENVIRONMENT NAME           *
                     * PACTSSYS:  SYSTEM NAME                *
                     * PACTSSBS:  SUBSYSTEM NAME             *
                     * PACTSTYP:  TYPE                       *
                     * PACTSSTG:  STAGE NAME                 *
                     * PACTSELM:  ELEMENT NAME               *
                     ****************************************/
      FIELD PANEL(BLG0C030) INDEX(S0E01)
       TEXT ('ACTION:  =PACTACTN  =PACTCOMM')
       TEXT ('ELEMENT: =PACTSENV / =PACTSSYS / =PACTSSBS / ') -
            ('=PACTSTYP / =PACTSSTG / =PACTSELMFN ');
    EPOPULATE;
ELABEL;
```

```
LABEL GRP0030;       /* FOR PACKAGE ACTIONS: RETRIEVE          */
  RC (0);            /*                                        */
  CRITERIA;
  ECRITERIA;
     POPULATE;
                     /*****************************************
                      * PACTACTN:  ACTION NAME                *
                      * PACTCOMM:  ACTION COMMENT             *
                      * PACTTDSN:  DATA SET NAME              *
                      * PACTTMBR:  MEMBER                     *
                      * PACTSENV:  ENVIRONMENT NAME           *
                      * PACTSSYS:  SYSTEM NAME                *
                      * PACTSSBS:  SUBSYSTEM NAME             *
                      * PACTSTYP:  TYPE                       *
                      * PACTSSTG:  STAGE NAME                 *
                      * PACTSELM:  ELEMENT NAME               *
                      *****************************************/
       FIELD PANEL(BLG0C030) INDEX(S0E01)
        TEXT ('ACTION:  =PACTACTN  =PACTCOMM')
        TEXT ('ELEMENT: =PACTSENV / =PACTSSYS / =PACTSSBS / ') -
             ('=PACTSTYP / =PACTSSTG / =PACTSELMFN ');
     EPOPULATE;
  CPASS GRP0070;
ELABEL;


LABEL GRP0040;       /* FOR PACKAGE ACTIONS: MOVE              */
  RC (0);            /*                      TRANSFER          */
  CRITERIA;
  ECRITERIA;
     POPULATE;
                     /*****************************************
                      * PACTACTN:  ACTION NAME                *
                      * PACTCOMM:  ACTION COMMENT             *
                      * PACTSENV:  ENVIRONMENT NAME           *
                      * PACTSSYS:  SYSTEM NAME                *
                      * PACTSSBS:  SUBSYSTEM NAME             *
                      * PACTSTYP:  TYPE                       *
                      * PACTSSTG:  STAGE NAME                 *
                      * PACTSELM:  ELEMENT NAME               *
                      * PACTTENV:  ENVIRONMENT NAME           *
                      * PACTTSYS:  SYSTEM NAME                *
                      * PACTTSBS:  SUBSYSTEM NAME             *
                      * PACTTTYP:  TYPE                       *
                      * PACTTSTG:  STAGE NAME                 *
                      * PACTTELM:  ELEMENT NAME               *
                      *****************************************/
       FIELD PANEL(BLG0C030) INDEX(S0E01)
```

```
        TEXT ('ACTION:  =PACTACTN  =PACTCOMM')
        TEXT ('SOURCE:  =PACTSENV / =PACTSSYS / =PACTSSBS / ') -
             ('=PACTSTYP / =PACTSSTG / =PACTSELMFN ')
        TEXT ('TARGET:  =PACTTENV / =PACTTSYS / =PACTTSBS / ') -
             ('=PACTTTYP / =PACTTSTG / =PACTTELMFN ');
     EPOPULATE;
ELABEL;

LABEL GRP0050;        /* ADD PATH  INFO FOR  ADD, UPDATE       */
  RC (0);             /*                                       */
  CRITERIA;
    FIELD (=PACTSFLAG) VALUE('P');
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG0C030) INDEX(S0E01)
        TEXT ('SOURCE:  =PACTSPANM  =PACTSFINM ');
     EPOPULATE;

  CFAIL GRP0060;
ELABEL;

LABEL GRP0060;        /* ADD DATA SET INFO FOR  ADD, UPDATE    */
  RC (0);             /*                        RESTORE        */
  CRITERIA;
    FIELD (=PACTSDSN) VALUE(NONBLANK);
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG0C030) INDEX(S0E01)
        TEXT ('SOURCE:  =PACTSDSN  =PACTSMBR ');
     EPOPULATE;
ELABEL;

LABEL GRP0070;        /* ADD PATH  INFO FOR  RETRIEVE          */
  RC (0);             /*                                       */
  CRITERIA;
    FIELD (=PACTTFLAG) VALUE('P');
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG0C030) INDEX(S0E01)
        TEXT ('TARGET:  =PACTTPANM  =PACTTFINM ');
     EPOPULATE;

  CFAIL GRP0080;
ELABEL;

LABEL GRP0080;        /* ADD DATA SET INFO FOR  RETRIEVE       */
  RC (0);             /*                        RESTORE        */
  CRITERIA;
    FIELD (=PACTTDSN) VALUE(NONBLANK);
```

```
  ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG0C030) INDEX(S0E01)
        TEXT ('TARGET:  =PACTTDSN  =PACTTMBR ');
     EPOPULATE;
ELABEL;
EUSE;


/*********************************************
 * EXIT POINTS: REVIEW/AFTER  EXECUTE/AFTER     *
 *              SHIP/AFTER    CONFIRM/AFTER     *
 *              BACKOUT/AFTER BACKIN/AFTER      *
 *              COMMIT/AFTER                    *
 *********************************************/

USE ((REVIEW, AFTER),
     (EXECUTE, AFTER),
     (SHIP, AFTER),
     (CONFIRM, AFTER),
     (BACKOUT,AFTER),
     (BACKIN,AFTER),
     (COMMIT,AFTER));

LABEL GRP0010;       /* MAINLINE - ALL POINTS                */
   RC(0);
   CRITERIA;
   ECRITERIA;
     POPULATE;
                    /*****************************************
                     * PREQCOMM:  PACKAGE COMMENT            *
                     * PREQEWSD:  EXECUTION WINDOW START DATE *
                     * PREQEWST:  EXECUTION WINDOW START TIME *
                     * PREQEWED:  EXECUTION WINDOW END DATE   *
                     * PREQEWET:  EXECUTION WINDOW END TIME   *
                     * PECBUSER:  CURRENT USER ID             *
                     * PECBFNNM:  PACKAGE FUNCTION            *
                     * PECBBANM:  BEFORE OR AFTER             *
                     * PECBNDRC:  NDVR HIGH RETURN CODE       *
                     * PHDRSTAT:  PACKAGE STATUS              *
                     *****************************************/
       FIELD PANEL(BLG6DSAB) INDEX(S0E0F) VALUE(=PREQCOMM);
       FIELD PANEL(BLG6SCHD) INDEX(S0C41) VALUE(=PREQEWSD);
       FIELD PANEL(BLG6SCHT) INDEX(S0C6E) VALUE(=PREQEWST);
       FIELD PANEL(BLG6TARD) INDEX(S0C42) VALUE(=PREQEWED);
       FIELD PANEL(BLG6TART) INDEX(S0C6F) VALUE(=PREQEWET);
       FIELD PANEL(BLG600CN) INDEX(S0B5C) VALUE(=PECBUSER);
       FIELD PANEL(BLG6PHAC) INDEX(S0C0B) VALUE(=PECBFNNM);
       FIELD PANEL(BLG0C010) INDEX(S0E01)
```

```
        TEXT ('PACKAGE PROCESSING COMPLETED RC = =PECBNDRC')
        TEXT ('INFO ACCESSED FROM PACKAGE EXIT =PECBBANM / =PECBFNNM')
        TEXT ('PKG STATUS: =PHDRSTAT  USER: =PECBUSER');
     EPOPULATE;

   CPASS EXE0010;
ELABEL;

LABEL EXE0010;       /* EXECUTE AFTER - LOG SECTION           */
  CRITERIA;
    FIELD (=PECBFNNM) VALUE ('EXECUTE ');
  ECRITERIA;
     POPULATE;
                    /*****************************************
                     * PHDRXD:    EXECUTION DATE             *
                     * PHDRXT:    EXECUTION TIME             *
                     * PHDREEXD:  END EXECUTION DATE         *
                     * PHDREEXT:  END EXECUTION TIME         *
                     *****************************************/
       FIELD PANEL(BLG0C010) INDEX(S0E01)
        TEXT ('EXECUTION:     =PHDRXD  =PHDRXT')
        TEXT ('END EXECUTION: =PHDREEXD  =PHDREEXT');
     EPOPULATE;

     CFAIL BKO0010;
ELABEL;

LABEL BKO0010;       /* BACKOUT AFTER - LOG SECTION           */
  CRITERIA;
    FIELD (=PECBFNNM) VALUE ('BACKOUT ');
  ECRITERIA;
     POPULATE;
                    /*****************************************
                     * PHDRBOST:  BACKOUT STATUS             *
                     * PHDRBOD:   BACKOUT DATE               *
                     * PHDRBOT:   BACKOUT TIME               *
                     *****************************************/
       FIELD PANEL(BLG0C010) INDEX(S0E01)
        TEXT ('BACKOUT:       =PHDRBOD  =PHDRBOT  STATUS: =PHDRBOST');
     EPOPULATE;

     CFAIL BKI0010;
ELABEL;

LABEL BKI0010;       /* BACKIN  AFTER - LOG SECTION           */
  CRITERIA;
    FIELD (=PECBFNNM) VALUE ('BACKIN  ');
  ECRITERIA;
     POPULATE;
```

```
                    /****************************************
                     * PHDRBID:   BACKIN DATE                *
                     * PHDRBIT:   BACKIN TIME                *
                     ****************************************/
       FIELD PANEL(BLG0C010) INDEX(S0E01)
        TEXT ('BACKIN:        =PHDRBID  =PHDRBIT');
     EPOPULATE;

     CFAIL COM0010;
ELABEL;

LABEL COM0010;       /* COMMIT  AFTER - LOG SECTION           */
  CRITERIA;
    FIELD (=PECBFNNM) VALUE ('COMMIT  ');
  ECRITERIA;
     POPULATE;
                    /****************************************
                     * PHDRCMD:   COMMIT DATE                *
                     * PHDRCMT:   COMMIT TIME                *
                     ****************************************/
       FIELD PANEL(BLG0C010) INDEX(S0E01)
        TEXT ('COMMIT:        =PHDRCMD  =PHDRCMT');
     EPOPULATE;

     CFAIL SHP0010;
ELABEL;


LABEL SHP0010;       /* SHIP XMIT AFTER - LOG SECTION         */
  CRITERIA;
    FIELD (=PECBSFNM) VALUE ('XMIT    ');
  ECRITERIA;
     POPULATE;
                    /****************************************
                     * PREQDEST:  SHIP DESTINATION           *
                     * PREQSTYP:  SHIPMENT TYPE              *
                     * PREQSCMP:  SHIP COMPLEMENTS (Y/N)     *
                     ****************************************/
       FIELD PANEL(BLG0C010) INDEX(S0E01)
        TEXT ('SHIP:   DESTINATION =PREQDEST  TYPE =PREQSTYP  ') -
             ('COMPLEMENTS =PREQSCMP');
     EPOPULATE;

     CFAIL CFM0010;
ELABEL;

LABEL CFM0010;       /* SHIP CONFIRM AFTER - LOG SECTION      */
  CRITERIA;
    FIELD (=PECBSFNM) VALUE ('CONFIRM ');
```

```
  ECRITERIA;
     POPULATE;
                    /*****************************************
                     * PREQDEST:  SHIP DESTINATION           *
                     * PREQSCNF:  SHIP CONFIRMATIN TYPE      *
                     * PREQSRES:  SHIP CONFIRMATION RESULTS  *
                     * PREQSRCV:  SHIP CONFIRM RC VALUE      *
                     *****************************************/
       FIELD PANEL(BLG0C010) INDEX(S0E01)
        TEXT ('SHIP:   DESTINATION =PREQDEST  CONFIRM TYPE =PREQSCNF') -
             ('        CONFIRMATION RESULTS  =PREQSRES =PREQSRCV');
     EPOPULATE;

ELABEL;

EUSE;


/***********************************************
 * EXIT POINTS: ADD/BEFORE     UPDATE/BEFORE    *
 *              RESTORE/BEFORE RETRIEVE/BEFORE  *
 *              ARCHIVE/BEFORE GENERATE/BEFORE  *
 *              DELETE/BEFORE  MOVE/BEFORE      *
 *              TRANSFER/BEFORE                 *
 ***********************************************/

USE ((ADD, BEFORE),
     (UPDATE, BEFORE),
     (RESTORE, BEFORE),
     (RETRIEVE, BEFORE),
     (ARCHIVE, BEFORE),
     (GENERATE, BEFORE),
     (DELETE, BEFORE),
     (MOVE, BEFORE),
     (TRANSFER, BEFORE));

NOTFOUND (RETURN);
FOUND (RETURN);      /* GREAT -- JUST RETURN                  */

EUSE;


/***********************************************
 * ON PACKAGE ACTIONS - JUST BEFORE EXECUTION. *
 * PURPOSE:  RECORD ACTION RETURN CODE ON       *
 *           THE CCID ACTIVITY RECORD.         *
 ***********************************************/

USE ((ADD, AFTER),
```

```
     (UPDATE, AFTER),
     (RESTORE, AFTER),
     (RETRIEVE, AFTER),
     (ARCHIVE, AFTER),
     (GENERATE, AFTER),
     (DELETE, AFTER),
     (MOVE, AFTER),
     (TRANSFER, AFTER));

NOTFOUND (ERR0010);
FOUND (GRP0010);

LABEL GRP0010;       /* ALL PKG ACTION POINTS                */
   RC(0);
   CRITERIA;
   ECRITERIA;
     POPULATE;
       FIELD PANEL(BLG0C030) INDEX(S0E01)
        TEXT ('ACTION: =ACTION  =DATE =TIME  USER: =USER  RC: =ACTRC')
        TEXT ('COMMENT: =COMM');
     EPOPULATE;

ELABEL;

LABEL ERR0010;       /* TELL USER                            */
  MSG ('ACTIVITY CCID =CCID RECORD NOT FOUND');
ELABEL;

EUSE;


/*********************************************
 * EXIT POINT:  DELETE AFTER                 *
 * PURPOSE:  DELETE CHANGE AND ACTIVITY PKG  *
 *           RECORDS, THEN RETURN.           *
 *********************************************/

USE (PDELETE, AFTER);

NOTFOUND (RETURN);
FOUND (DELETE,RETURN);

EUSE;

ETABLE;
```

Chapter 7: Displaying Interface Information

This section contains the following topics:

How to Display Information Stored in InfoMan

The CA Endevor InfoMan Interface allows users, in foreground, to:

- Produce selection lists of CCIDs when requesting stand-alone actions.
- Display the information stored in InfoMan about a CCID.
- Display the information stored in InfoMan about a CA Endevor SCM package.

The InfoMan record display panel is the same for packages and for CCIDs. The information displayed is basic and some fields may be blank when they are not used in a site's InfoMan policy.

How to List and Display CCID Information

When the CA Endevor InfoMan Interface is active, users can produce a list of available CCIDs by wildcarding the CCID field. The CCID list can be produced when requesting the following stand-alone actions in foreground and when generating SCL for execution in batch.

- ADD/UPDATE
- MOVE
- RETRIEVE
- GENERATE
- DELETE
- TRANSFER
- ARCHIVE

The interface searches the Change and Problem records in InfoMan and returns a list of CCIDs that have an InfoMan type of ACTION, an InfoMan status other than CLOSED, and meet the wildcard criteria.

Note: Wildcarding CCIDs is not allowed when building packages. When a CCID value is fully specified, a list is not produced. If no InfoMan records meet the wildcard criteria, a CCID prompt panel appears. The user can type either a fully specified CCID or a wildcarded CCID on this panel. This search process can be repeated multiple times if necessary.

From the CCID list, you can either view InfoMan correlation information for one or more CCIDs on the list, or select one of the CCIDs for the action being requested.

Important! If you use the CA Endevor InfoMan Interface, make sure you set up all your CA Endevor SCM systems to require CCIDs. For more information, see the *Administration Guide*.

When using the CA Endevor InfoMan Interface, CA Endevor SCM CCIDs must start with an alpha character. In addition, InfoMan stores only the first eight characters of the CA Endevor SCM CCID.

The CCID List Panel

The CCID List panel appears when you specify a full or partial wildcard in the CCID field on an action request panel. The CA Endevor SCM location fields on this panel display the current environment, system, subsystem, type, and stage, and the element against which the action is being requested. Other panel components are described below.

S--SELECT

Type this option next to the desired CCID to choose that CCID from this list.

B--BROWSE

Use this option to tell CA Endevor SCM to display the InfoMan record associated with this CCID.

RNID/CCID

This field displays InfoMan record IDs (RNIDs).

RECORD DESCRIPTION

This field displays the description associated with the InfoMan RNID.

The Action Prompt Panel

When no CCIDs on the InfoMan side meet the wildcard criteria and the system named on the panel has been defined to require CCIDs, the Action Prompt panel is displayed, allowing the user to enter a new CCID value.

The CCID provided on the prompt panel can also be wildcarded. If CCIDs on the InfoMan side meet this wildcard criteria, a list selection list appears.

After a user selects a CCID, the interface returns the original action panel.

The CCID Correlation Panel

When you type **B** (Browse) next to a CCID on the CCID List panel, and InfoMan contains information about that CCID, the CCID correlation panel appears when you press Enter.

From this panel you can:

- Press PF3 to return to CA Endevor SCM.
- Press PF1 to access help for this panel.

How to Display Package Information

When package information exists in the InfoMan database, an additional option, CI - Display Correlation Information, appears on the following panels:

- Package Display
- Create/Modify Package
- Cast Package
- Review Package
- Execute/Submit Package
- Backout Package
- Commit Package (appears as I - Display Correlation Information)

If your site does not use InfoMan, or InfoMan correlation information does not exist, this option does not appear on the panel.

The CI option behaves as follows.

- The option appears on the appropriate panels if InfoMan is installed, regardless of whether InfoMan is storing any package information.
- If no correlation exists, the option and the text are shown in normal intensity.

- If correlation information exists, the option is shown in high intensity when certain conditions are met. These conditions are described below.
- If the user selects the CI option when no correlation information exists, the ISPF short message INVALID OPTION appears after you press Enter.

The CI option activates in either of two ways.

- If the user creates the parent record in InfoMan using the CA Endevor SCM scripts, then CI turns to high intensity when the package is cast.
- If the user creates the parent record directly in InfoMan, then CI turns to high intensity at the first exit point invoked from CA Endevor SCM after the Cast action.

This means that the CI option may stay at low intensity, and therefore unavailable, even when package information exists in InfoMan. For example, if a user creates parent records for packages directly in InfoMan, and builds an E/INFO table with only a (CREATE,AFTER) USE statement, the CI option will not turn to high intensity.

When you type CI and press Enter, this correlation panel appears. Press PF3 to return to the package panel from which you accessed the correlation information.

Chapter 8: Using Interface Reports

This section contains the following topics:

The Interface Reports

The CA Endevor InfoMan Interface reports are designed to show the status of records in InfoMan and to show their relationship to CA Endevor SCM information. The reports provide an audit trail for packages, package actions, and CCIDs. The reports are built using the InfoMan Report Format Table (RFT) facility.

The following table contains the RFT streams that produce each report.

This RFT stream	Produces this interface report
EINTACH0	Action Change Record Summary
EINTACH1	Action Change Record Detail
EINTAPB0	Action Problem Record Summary
EINTAPB1	Action Problem Record Detail
EINTPKG0	Package Parent Record Summary
EINTPKG1	Package/Activity Record Summary
EINTPKG2	Package/Activity Record Detail

These RFT streams are located in iprfx.iqual.CSIQOPTN.

Run the Reports in Native InfoMan

To run the reports in native InfoMan, type the following on the InfoMan command line, then press Enter:

```
REPORT,8,RFT name
```

In this syntax, RFT name can be any of the RFT streams in the table in the preceding section. For example, the following command generates the Action Change Record Summary:

```
REPORT,8,EINTACH0
```

InfoMan prompts you for the report destination if the session defaults for print, report, and for customized report destination are not in your InfoMan profile. To update these fields in your profile select option **2** (PROFILE) from the InfoMan Main Menu to set the values, or contact your site InfoMan administrator.

Run the Reports in Batch

The user submitting batch report jobs must be defined as a user to InfoMan. There are two additional issues that you must address before running interface reports in batch.

- Contention for the ISPF profile data set.
- Setting defaults in InfoMan for output destination and class for reports.

Contention for the ISPF Profile Data Set

InfoMan requires access to the ISPF profile data set when executing report requests in batch. This means that contention occurs when you use the same ISPF profile data set to run reports in batch and to have an active ISPF/TSO session. To resolve this problem, copy your interactive ISPF profile data set and rename it userid.iqual.ISPPROF. Use the renamed profile for use when logging on in batch.

How to Set Defaults in InfoMan for Output Destination and Class for Reports

InfoMan stores user profile information, including default output destination and class information for reports, in the BLG0PROF member in the ISPF profile data set.

These defaults can be set in either of two ways:

- Each user can set their own options by invoking an interactive InfoMan session and updating their own BLG0PROF profile member.
- The InfoMan administrator can set up a default profile member for all users.

To run the reports in batch, modify JCL member BC1JEI35 in the iprfx.iqual.CSIQJCL data set. The default input stream is all reports. You can select the reports you want to run by modifying the input stream.

How to Customize the Reports

You can customize the standard reports, or create your own reports using InfoMan's RFT facility.

Note: For more information about the InfoMan RFT facility, see your InfoMan administrator.

The Action Change Record Summary Report

The Action Change Record Summary report lists the InfoMan Change records recorded as type ACTION. The list is sorted by CCID.

Use this report as an inventory list of CCIDs, and to monitor the status of each CCID.

The following describes the report fields.

CCID

The CA Endevor SCM CCID for the action, as recorded in the InfoMan RNID field.

Description of InfoMan Change Record

The description of the action from the Populate block of the batch utility, recorded in the freeform text field in the InfoMan record.

Last Mod Userid

The ID of the user who last modified the InfoMan record.

Last Mod Date

The date of the last modification.

Last Mod Time

The time of the last modification.

Change Status

Change status of the action.

Approval Status

Approval status of the action.

The Action Change Record Detail Report

The Action Change Record Detail report prints one page for each InfoMan Change record listed on the Action Change Detail Summary report. In addition to repeating the information in the summary report, this report prints any freeform text associated with the Change record. There is one freeform text entry for each action associated with the CCID.

Use this report as an audit trail of the actions associated with a particular CCID.

The Action Problem Record Summary/Detail Reports

The Action Level Interface Problem Record Summary and Detail reports contain information similar to that in the Change record reports.

Use these reports the same way that you would use the Action Change Record reports: the Summary report as an inventory list for CCIDs, and the Detail report as an audit trail for actions associated with CCIDs.

The following describes the report fields.

CCID

The CA Endevor SCM CCID. For action problem records, this is the same as the InfoMan RNID.

Description

The description of the action from the Populate block of the batch utility, recorded in the free form text field in the InfoMan record.

Last Mod Userid

The ID of the user who last modified the InfoMan record.

Last Mod Date

The date of the last modification.

Last Mod Time

The time of the last modification.

Problem Status

The status of the problem.

Date entered

The date this problem record was created in InfoMan.

Assignee name

The person who has been assigned the problem.

Free form text

Appears on the detail report only. Displays information about this action.

The Package Parent Record Summary Report

The Package Level Interface Parent Record Summary report lists InfoMan Change records with a type of PACKAGE.

The following describes the report fields.

Curr Stat

Current status of the package change record.

InfoMan RNID

The InfoMan RNID associated with this change record. The RNID is cross referenced with the correlation field in the package header in CA Endevor SCM.

CA Endevor SCM package

The ID of the package in CA Endevor SCM. The package ID is cross referenced with the InfoMan co-requisite field.

InfoMan update

The date when this InfoMan record was last updated.

User

The ID of the user who last updated the record.

Package execution window - start

The date and time of the start of the package execution window.

Package execution window - end

The date and time of the end of the package execution window.

Approval status

User-defined status of the approval process.

Description

Description of the package change record.

The Package/Activity Record Summary Report

The Package/Activity Record Summary report lists package parent record summary information along with summary information for any associated package actions. The report is sorted by the package parent record InfoMan RNID.

This report contains the following fields:

Package Information

This report contains the following package information:

Curr Stat

Current status of the package change record.

INFO RNID

The InfoMan RNID associated with this change record. The RNID is cross referenced with the CORRELATION field in the package header in CA Endevor SCM.

CA Endevor SCM package

The ID of the package in CA Endevor SCM. The package ID is cross referenced with the InfoMan COREQUISITE field.

InfoMan update

The date when this InfoMan record was last updated.

User

The ID of the user who last updated the record.

Package execution window - start

The date and time when the package execution window starts.

Package execution window - end

The date and time of the end of the package execution window.

Approval status

User-defined status of the approval process.

Package status

Current status of the package in CA Endevor SCM.

Last package function

Last package function executed in CA Endevor SCM.

Description

Description of the package.

Activity Information

This report contains the following activity information:

RNID

RNID associated with this activity record.

CCID

CA Endevor SCM CCID associated with this activity record.

Last update - date

Date of the last update.

Last update - user

ID of the user who last updated this record.

Activity type

Always PKG/ACT for activity records.

Description

Description of the activity record.

The Package/Activity Record Detail--Package Detail Report

The Package/Activity Record Detail report provides detail information for both the package parent record and any associated activity records. Detail information includes the summary information plus any freeform text associated with the parent package record or the activity records.

Page 1 of the report contains the information for the parent package record. The report starts a new page for each activity record.

The Package/Activity Detail--Activity Detail Report

The activity detail portion of the Package Activity Detail report starts a new page for each activity record associated with the parent change record. The information includes the same information as on the summary report, and includes freeform text associated with each activity record.

The freeform text of activity records contains one entry for each action associated with the activity record CCID.

Appendix A: Information for Advanced Users

This section contains the following topics:

The Modified Assisted Entry Panel (BLG6CORQ)

The BLG6CORQ Panel provided by standard InfoMan must be modified to allow it to store the 16-character CA Endevor SCM CCID.

Phase 2, Step 3 of the installation procedure documented in Step 3. Copy Dictionary Update explains how to modify the BLG6CORQ panel if you are working with standard InfoMan.

InfoMan administrators at sites that have already modified their InfoMan panels should do the following:

1. Create a PWORD of RNCC/ with a validation pattern of CCV15.
2. Assign an available prefix index number to this PWORD.
3. Update the BLG6CORQ panel to point to the index number you have assigned

The reports shown later in this section are standard InfoMan reports that provide the information needed to modify the BLG6CORQ Panel to support the interface.

The PMF Report for the BLG6CORQ Panel

This report is for the use of an InfoMan expert at your site when modifying the BLG6CORQ Panel. The index value must be assigned by the InfoMan administrator.

Customized Dictionary Entry

In addition to modifying the BLG6CORQ Panel, you also must add an entry to the dictionary data set specified in the BLGSESxx session parameters member.

The second report in this section is a PWORD XREF report, and contains the PWORD dictionary entry for the modified Assisted Entry panel.

This report is for the use of the InfoMan expert at a site that has already customized InfoMan. The report provides the InfoMan technician with the information needed to customize the appropriate dictionary entry.

The PWORD XREF Report

The index value in the PWORD XREF report must be assigned by the InfoMan administrator.

Reserved InfoMan Fields

The fields summarized on the following pages are reserved by the interface for its use. Make sure that these fields are not being used to store any other data, because the interface overwrites any data it finds in these fields.

Inquiry PIDT Required Fields

Inquiry against Change records, for packages (IPKG).

This PIDT field	Is reserved for this information
FIELD PANEL (BLG00000) INDEX(S0B06)	Change type record
FIELD PANEL (BLG6CORQ) INDEX(S0CD4)	Package ID
FIELD PANEL (BLG6PTYP) INDEX(S0C09)	Type: PACKAGE
FIELD PANEL (BLG6ACCN) INDEX(S0CCF)	RNID value

Inquiry against Activity records, for package actions (IPKGA).

This PIDT field	Is reserved for this information
FIELD PANEL (BLG0F000) INDEX(S0B07)	Activity type record
FIELD PANEL (BLG6RNOR) INDEX(S0CD0)	Parent RNID

This PIDT field	Is reserved for this information
FIELD PANEL (BLG6ACNM) INDEX(S0CBC)	CCID
FIELD PANEL (BLG6PTYP) INDEX(S0C09)	Type: PKG/ACT
FIELD PANEL (BLG6ACCN) INDEX(S0CCF)	RNID value

Inquiry against Change records, for actions (IACT).

This PIDT field	Is reserved for this information
FIELD PANEL (BLG00000) INDEX(S0B06)	Change type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: ACTION
FIELD PANEL(BLG6ACCN) INDEX(S0CCF)	RNID value

Inquiry against Problem records, for actions (IACT).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0032)	Problem type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: ACTION
FIELD PANEL(BLG6ACCN) INDEX(S0CCF)	RNID value

Retrieve PIDT Required Fields

Retrieve against Change records, for packages (RPKG).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0B06)	Change type record
FIELD PANEL(BLG6CORQ) INDEX(S0CD4)	Package ID
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: PACKAGE

Retrieve against Activity records, for package actions (RPKGA).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG1A131) INDEX(S0B07)	Activity type record
FIELD PANEL(BLG6RNOR) INDEX(S0CD0)	Parent RNID

This PIDT field	Is reserved for this information
FIELD PANEL (BLG6PTYP) INDEX(S0C09)	Type: PKG/ACT
FIELD PANEL(BLG6ACNM) INDEX(S0CBC)	CCID

Retrieve against Change records, for actions (RACT).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0B06)	Change type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: ACTION

Retrieve against Problem records, for actions (RACT).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0032)	Problem type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: ACTION

Create PIDT Required Fields

Create Change records for packages (CPKG).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0B06)	Change type record
FIELD PANEL(BLG6CORQ) INDEX(S0Cd4)	Package ID
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: PACKAGE
FIELD PANEL(BLG6URN0) INDEX(S0CCF)	RNID value

Create Activity records for package actions (CPKGA).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG1A131) INDEX(S0B07)	Activity type record
FIELD PANEL(BLG6RNOR) INDEX(S0CD0)	Parent RNID
FIELD PANEL(BLG6ACNM) INDEX(S0CBC)	CCID
FIELD PANEL (BLG6PTYP) INDEX(S0C09)	Type: PKG/ACT

This PIDT field	Is reserved for this information
FIELD PANEL(BLG6URN0) INDEX(S0CCF)	RNID value

Create Change records for actions (CACT).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0B06)	Change type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: package or action
FIELD PANEL(BLG6URN0) INDEX(S0CCF)	RNID value

Create Problem records for actions (CACT).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0032)	Change type record
FIELD PANEL(BLG6PTYP) INDEX(S0C09)	Type: package or action
FIELD PANEL(BLG6URN0) INDEX(S0CCF)	RNID value

Update PIDT Required Fields

Update Change records for packages (UPKG).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0B06)	Change type record

Update Activity records for package actions (UPKGA).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG1A131) INDEX(S0B07)	Activity type record
FIELD PANEL (BLG6PTYP) INDEX(S0C09)	Type: PKG/ACT

Update Change records for actions (UACT).

This PIDT field	Is reserved for this information
FIELD PANEL(BLG00000) INDEX(S0B06)	Change type record

Update Problem records for actions (UACT).

This PIDT field		Is reserved for this information
FIELD PANEL(BLG00000)	INDEX(S0032)	Problem type record

Appendix B: Installing the Interface Under IIF

This section contains the following topics:

How to Install the CA Endevor InfoMan Interface Under IIF

The installation of the interface falls into four phases:

- Phase 1: Installing basic InfoMan.
- Phase 2: Modifying basic InfoMan to support the interface.
- Phase 3: Tying the CA Endevor SCM and InfoMan sides together.
- Phase 4: Verifying the installation

The target audience for this section is those people who have a strong understanding of InfoMan IIF records and a general knowledge of CA Endevor SCM. The best combination of people to install the interface is the site InfoMan administrator and the site CA Endevor SCM administrator working together.

Phase 1. How to Install Basic InfoMan

The requirements in this section are basic InfoMan installation requirements that exist whether or not the interface is installed.

Note: CA Endevor InfoMan Interface requires that the IBM Information Management product, version 4.2 or higher, is installed at your site.

InfoMan setup involves the following.

1. Define all interface users to a class in InfoMan.
2. Make sure the Information Management LOAD LIBrary data is in LINKLST or that you have its data set name available.
3. Prepare a session parameter (BLGSESxx) module to reflect the database, data dictionary, read panel, and RFT data sets that the interface uses when accessing the data base via the API.
4. Make sure that BLGISPFD, an InfoMan supplied ISPF panel, has been moved into your ISPPLIB concatenation.

Step 1. Define Users to InfoMan Classes

Each person authorized to use the interface must be defined to a class in InfoMan. See the InfoMan documentation for instructions.

Step 2. Prepare a Session Parameter Member

InfoMan requires a BLGSESxx module that points to the database, data dictionary, read panels, and RFT data set that the Interface uses when accessing the database via the API or native InfoMan.

This session parameter is well documented in the InfoMan install manuals. It is required of standard InfoMan. The naming standard is BLGSESxx where xx is the user's choice. This suffix is also required for a CA Endevor SCM start-up block assembly. Note that the same BLGSESxx member can be used for the interface as well as standard access to InfoMan.

The suffix used to name the actual BLGSESxx module is required in Phase 3. The rest of this install requires the data set names from the BLGSESxx module.

Note: The interface installation includes 15 modified panels, which must be copied into your read panel data set concatenation. You can either overlay the IBM version of this data set, or provide a new read panel data set ahead of the standard IBM data set. Either way, the result is that the RPANEL0 label points to the data set used in the install.

Step 3. Put Information Management Load Data Set in LINKLIST

The InfoMan load data set is used to install the interface.

Sometimes this library is part of LINKLIST, and sometimes it is not. If its data set is not in LINKLST, the person implementing the interface needs to know the name, and must put it into a STEPLIB in the concatenation.

Step 4. Move BLGISPFD into ISPPLIB

Make sure that BLGISPFD, an InfoMan-supplied ISPF panel, has been moved into your ISPPLIB concatenation. See your InfoMan administrator if you have any questions about this step.

Phase 1 Summary

In this phase of the implementation you have built a vanilla version of InfoMan IIF. At this point you should have installed InfoMan on your system, assigned users to an InfoMan class, and defined a BLGSESxx session parameter module. For future reference, record the following:

BLGSESxx:

Within BLGSESxx, you have defined:

```
Dictionary DD name:

RFTDS DS name:

RPANEL0 name:
```

INFOMAN LOAD DS name:

Phase 2. How to Modify Basic InfoMan for the Interface

In this phase, you customize the base InfoMan system set up in Phase 1 by performing the following steps

1. Copy interface report shells to the RFT data set identified in Phase 1.

   BC1JEI20

2. Copy the CA Endevor SCM versions of the InfoMan IIF panels into the RPANEL0 data set identified in Phase 1.

   BC1JEX15

3. Install the dictionary updates.

   BC1JEI30

4. Create the IIF PIDT and PIPT tables.

   BC1JEX45

5. Build a rule set for driving the API using the interface batch utility.

   BC1JEX00

6. Assemble or copy the BLGSESxx module into your STEPLIB data set.

   BC1JEI05

Step 1. Copy Interface Report Shells into the RFT Data Set

Copy the interface report shells (RFTs) to your RFT data set. Do this using the BC1JEI20 member in iprfx.iqual.CSIQJCL. Before submitting this JCL, provide a job card, confirm that *iprfx.iqual* and uprfx.uqual are the correct data set qualifiers, and confirm that tdisk is a valid unit name.

The BC1JE120 job copies the following report shells

EINTACH0

Action Change Record Summary

EINTACH1

Action Change Record Detail

EINTAPB0

Action Problem Record Summary

EINTAPB1

Action Problem Record Detail

EINTPKG0

Package Parent Record Summary

EINTPKG1

Package/Activity Record Summary

EINTPKG2

Package/Activity Record Detail

Step 2. Copy Modified Panels into the RPANEL0 Data Set

The interface requires that a modified version of the Assisted Entry Panel (BLG6CORQ) be available in the RPANEL0 data set pointed to by the BLGSESxx session parameter member. The modification allows the BLG6CORQ panel to store the 16-character CA Endevor SCM package ID. This field has been flagged as a reserved field for the interface.

The modified panel and other CA Endevor SCM IIF tailored panels are provided with the interface. Copy them into the RPANEL data set using the JCL in member BC1JEX15 in the iprfx.iqual.CSIQJCL data set.

This job uses the BLGUT6F utility to copy a PDS member that has been created from a InfoMan panel using BLGUT6.

This panel must reside above the IBM panel of the same name. The person who runs the job to copy the modified panel needs only the data set name. You recorded the RPANEL0 data set name in the worksheet at the end of Phase 1.

Before submitting this JCL, provide a job card, and confirm that *iprfx.iqual* and uprfx.uqual are the correct data set qualifiers.

Step 3. Copy Dictionary Update

The modified Assisted Entry Panel (BLG6CORQ) has an associated dictionary entry. This entry must be copied into the dictionary identified in the Phase 1 Summary. Run JCL job BC1JEI30 to install this dictionary entry.

Important! Run this job only if you are installing basic InfoMan. If your site has customized InfoMan, see the appendix "InfoMan Information for Advanced Users (see page 141)" for instructions on updating the dictionary entry.

Before submitting this JCL, provide a job card, and confirm that *iprfx.iqual* and uprfx.uqual are the correct data set qualifier.

Step 4. Create CA PIDT and PIPT Tables

Create the CA Technologies PIDT and PIPT tables by executing the InfoMan BLGUT8 utility program using the CA supplied source PIDT tables that reside in iprfx.iqual.CSIQOPTN. This step is necessary to pick up the updated data dictionary entry. The tables are required by the interface to support batch and real-time execution, CCID list support and display services.

Use the BC1JEX45 member in iprfx.iqual.CSIQJCL. This JCL is shown next. Before submitting the JCL, provide a job card and ensure that *iprfx.iqual* and uprfx.uqual are the correct data set qualifiers.

Step 5. Build a Rule Set

In addition to setting up InfoMan to support the interface, you need to build a rule set to drive the InfoMan API. You do this using the batch utility provided by the interface.

Before proceeding, review the following:

- Setting Up the E/INFO Table (see page 29)
- The samples provided in iprfx.iqual.CSIQOPTN. There are three samples provided:

EISCRXX1

Batch utility input for stand-alone actions, using IIF InfoMan Change records.

EISCRXX2

Batch utility input for stand-alone actions, using IIF InfoMan Problem records.

EISCRXX3

Batch utility input for package functions and package actions under IIF InfoMan.

Next, create a rule set. It is advisable to keep it simple at first. Do this by taking one or two actions or package functions and writing utility syntax input for them. Then run the batch utility to create an initial rule set. The JCL for executing the utility to build the rule set is in member BC1JEX00 of iprfx.iqual.CSIQJCL.

A successful run of the utility produces a load module called by a name you specify. This load module name is used in Phase 3. Assemble this load module into a STEPLIB data set.

Remember:

- To change what you use the interface for is a simple run of the utility. It requires no further InfoMan or CA Endevor SCM work to enhance the rule set.
- If you point to any PIDT tables other than the defaults, verify that they reside in the RFTDS data set pointed to in your BLGSESxx member.

The JCL for executing the utility to build the rule set is in member BC1JEX00 of iprfx.iqual.CSIQJCL. Before submitting this JCL, provide a job card, confirm that *iprfx.iqual* and uprfx.uqual are the correct data set qualifiers, confirm that tdisk is a valid unit name and, if necessary, change the size of the WORK1 or WORK2 data sets.

Step 6. Copy the BLGSESxx Module to STEPLIB

Assemble or copy the BLGSESxx member into your STEPLIB data set. See iprfx.iqual.CSIQJCL(BC1JEI05) for sample JCL.

Before submitting this JCL, provide a job card, confirm that uprfx.uqual are the correct data set qualifiers, confirm that tdisk is a valid unit name, and replace xx in the string BLGSESxx with a session identifier.

Phase 2 Summary

The pieces are now in place for the interface on both the InfoMan and the CA Endevor SCM sides. In Phase 3, these pieces are connected to enable the interface.

Phase 3. How to Connect the Interface

You must complete the tasks described for Phase 1 and Phase 2 before starting the tasks described in this section.

This phase of the installation process involves setting up the InfoMan and CA Endevor SCM sides of the interface to work together, as described:

1. Modify the @EINFO macro.
2. Assemble and link BC1TEI90 into a LINKLIST or authorized library.
3. Assemble C1DEFLTS with InfoMan password.
4. Make sure that required panels are in the ISPPLIB data set.
5. Make sure that members INFO01 and PKMR02 are in your ISPMLIB data set.
6. Edit C1SB3000 skeleton JCL. These steps are described in the following sections.

Step 1. Modify the @EINFO Macro

Various parts of the interface access the user-assembled module BC1TEI90 to obtain user parameter information. BC1TEI90 is a load module used during exits 5, 2, 3, and 7 for start-up parameters, during display services, and during batch execution. The user must assemble this load module. Once assembled, this module should be linked into a LINKLIST or authorized library.

The @EINFO macro, supplied with the interface, provides input to BC1TEI90. The macro can be found embedded in member BC1JEI10 in iprfx.iqual.CSIQJCL. Edit this macro based on the information contained in the Phase 1 checklist and your site standards. The @EINFO macro is shown next.

```
@EINFO ENTRY=START,
     BLGSESS=00,
     APISESID=USERID,
     EPIDT=ENDEVOR,
     INVCLASS=MASTER,
     WAITTIME=300,
     RECTYPE=CHANGE,<tab>
@EINFO ENTRY=END
```

The following describes the entries in this macro:

ENTRY=START

Indicates the beginning of the BC1TEI90 table.

BLGSES=

InfoMan looks for session parameters in a module named BLGSESxx, where 'xx' is a two-character identifier for the module. This module is described in the InfoMan documentation. You recorded the components of this module during Phase 1 of this installation, as well as the two-character identifier for the module. When modifying the @EINFO macro supplied with the interface, replace the default value 00 with the value you specified in Phase 1.

The default is 00. This is the same as standard InfoMan.

APISESID=

The API requires a session ID during initialization. This session can be called USERID or can be a predefined InfoMan API ID.

The default is USERID. Leaving this default allows the same start-up module to be used by many people, with the interface substituting the user ID of the caller for the literal USERID.

If you include this keyword, you must provide a value.

EPIDT=

This keyword contains the name of the assembled E/INFO table produced by the batch utility. The default is Endevor.

If you include this keyword, you must provide a value.

INVCLASS=

InfoMan has its own security system. It is broken into classes with privileges. Every user or API ID is assigned to at least one CLASS. The API requires a class name.

The default is MASTER.

WAITTIME=

The API allows the application to define how long it will wait for a response from the InfoMan API server. If the user does not use this keyword, the default will be 300 seconds.

RECTYPE=

Users must select one kind of InfoMan record for recording information about stand-alone actions. They can use either Change or Problem records. Display services also need this information.

The default is CHANGE. If coded, it cannot be blank.

ENTRY=END

(Required) This statement denotes the end of the BC1TEI90 table.

Step 2. Assemble and Link BC1TEI90

Assemble and link BC1TEI90 using the JCL in member BC1JEI10 in iprfx.iqual.CSIQJCL. Before submitting this JCL, provide a job card, confirm that *iprfx.iqual* and uprfx.uqual are the correct data set qualifiers, and that tdisk is a unit name.

Step 3. Reassemble C1DEFLTS with the InfoMan Password

Include the InfoMan password in your C1DEFLTS table. The password parameter is in the TYPE=MAIN section of the defaults table, as shown.

```
C1DEFLTS TYPE=MAIN,                                          X
       ACCSTBL=,               ACCESS SECURITY TABLE         X
       ACMIDXUP=N,             CROSS-REFERENCE DATA UPDATE   X
       ACMROOT=,               ROOT DATABASE                 X
       ACMXREF=,               XREF DATABASE                 X
       APRVFLG=N,              APPROVAL PROCESSING (Y/N)     X
       ASCM=N,                 ASCM CONTROL OPTION           X
       BATCHID=0,              BATCH UID FROM JOBNAME/USER=  X
       CIPODSN=,                         CCID VALIDATION DSN X
       CUNAME='*** PUT YOUR COMPANY NAME HERE ***', (50 CHAR) X

       DB2=N,                  DB2 CONTROL OPTION            X
       ESSI=N,                 ESSI CONTROL OPTION           X

       INFO=N,                 INFOMAN CONTROL OPTION        X
       LIBENV=,                LIBRARIAN (LB), PANVALET (PV) X
       LIBENVP=N,              LIBRARIAN/PANVALET OPTION     X
       LIBRPGM=,               LIBRARIAN BATCH PROGRAM NAME  X
       LINESPP=60,             LINES PER PAGE                X
       MACDSN='iprfx.iqual.CSIQOPTN',    E/OS/390 SOURCE LIBRARY X
       PKGDSN='UPRFX.UQUAL.PACKAGE',      PACKAGE DATASET NAME X
       PKGTSO=N,               FOREGROUND PACKAGE EXEC (Y//N) X
       PDM=N,                  PDM CONTROL OPTION            X
       PKGSEC=,                PACKAGE SECURITY              X
       PKGCSEC=N,              PACKAGE CAST SECURITY (Y/N)   X
       PKGCVAL=0,              PKG COMPONENT VALIDATION (Y/O) X
       PROC=N,                 PROCESSOR OPTION              X
       RACFUID=,               E/OS/390 RACF USERID          X
       SITEID=0,               E/OS/390 SITE ID              X
       SMFREC#=0,              SMF RECORD NUMBER             X
       SPFEDIT=SPFEDIT,        DEFAULT PDS RESERVE           X
       SYSIEWL=SYSIEWLP,       DEFAULT PDS/LINK EDIT RESERVE X
       UIDLOC=(1,7),           UID/JOBNAME START/LENGTH POS  X
       VIOUNIT=TDISK,          UNIT FOR VIO-ELIGIBLE ALLOC   X
       WRKUNIT=TDISK,          UNIT NAME FOR WORK SPACE      X
       WORKVOL=                VOL SER NUMBER FOR WRKUNIT
```

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 will store the defaults table in *iprfx.iqual.CSIQAUTU* as member C1DEFLTS.

Step 4. Make Sure Required Panels Are in ISPPLIB

Make sure that the panels listed next reside in your ISPPLIB. These panels can be found in iprfx.iqual.CSIQPENU.

C1EILIST, CITILIST

CCID list panel and its associated tutorial panel.

C1SEIBRW, CITEIBRW

InfoMan record display for packages and stand-alone actions and its associated tutorial panel.

C1SP1000, C1SP2000, C1SP3000, C1SP4000, C1SP5000, C1SP6000, C1SP7000

Package panels with capability to display correlation data.

CITP1000, CITP2000, CITP3000, CITP4000, CITP5000, CITP6000, CITP7000

Tutorials for package panels with capability to display correlation data.

Step 5. Make Sure INFO01, PKEX21, and PKMR02 Are in ISPMLIB

Members INFO01, PKEX21. and PKMR02 contain interface messages, and must reside in your ISPMLIB. These members can be found in *iprfx.iqual.CSIQMENU*.

Step 6. Edit C1SB3000 Skeleton JCL

Member C1SB3000 is used to submit CA Endevor SCM package and batch processing requests. The member can be found in *iprfx.iqual.CSIQSENU*. Before invoking this JCL, confirm that *iprfx.iqual* are the correct data set qualifiers, and that tdisk is a valid unit name.

Phase 4. How to Verify the Installation

The list next shows the members that should reside in the designated data sets for the interface to function as designed. If any of them are not there, the interface will not work as designed.

CSIQAUTH/CSIQAUTU

Load modules, as well as the following:

- C1BMEI00
- C1DEFLTS table. See Step 3. Reassemble C1DEFLTS with the InfoMan Password.
- C1SM1000
- BC1TEI90, the module containing user parameters, which you assembled in Phase 3, Step 1.
- E/INFO table. This is the rule set that you build using the batch utility. See Step 5. Build a Rule Set.
- BLGSESxx, the module containing the InfoMan session parameters. See Step 2. Prepare a Session Parameter Member.

CONLIB

The CA Endevor SCM load modules installed from *iprfx.iqual.CSIQLOAD*. Make sure to save the current copy of C1BEXITS before installing the new interface. This allows you to go back to the old interface during testing.

RFTDS

EINTACH0, EINTACH1, EINTAPB0, EINTAPB1.

EINTPKG0, EINTPKG1, EINTPKG2.

PIDT and PIPT tables. See the list in the next section. These panels are found in iprfx.iqual.CSIQOPTN.

BLGISPFD, the InfoMan supplied ISPF panel.

RPANEL0

BLG6CORQ, the modified version of the Assisted Entry panel.

BLGAPI10, BLGAPI90, BLGAPI92, BLGAPI95, BLGAPI99.

BLGZAACP, BLGAAUP.

BTN0B100, BTN0C100, BTN0C101, BTN0L100, BTN0M100, BTN0S020, BTN6PTY1.

See Step 3. Copy Dictionary Update.

DDICT

index RNCC/CCV15. This is the modified dictionary entry associated with the modified Assisted Entry Panel. See Step 4. Create the Standard IBM PIDT and PIPT Tables.

iprfx.iqual.CSIQPENU

C1EILIST, C1SEIBRW, C1SP1000 — C1SP7000 and the associated tutorials (CITILIST, CITEIBRW, and CITP1000-CITP7000.

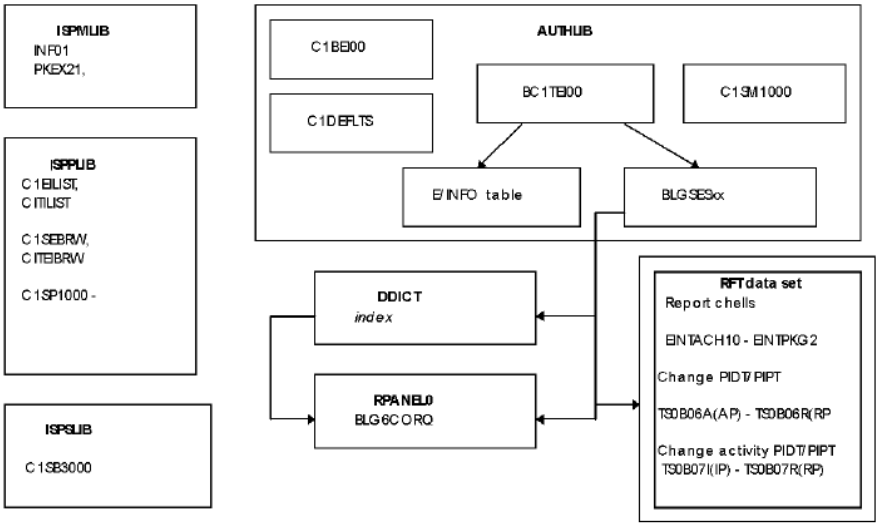
iprfx.iqual.CSIQMENU

INFO01, PKEX21, PKMR02.

iprfx.iqual.CSIQSENU

C1SB3000.

The graphic displayed next shows how these modules tie the interface together and indicates the important members that should be at each location.



As previously illustrated, the interfaces work based on how the modules tie them together.

Required PIDT and PIPT Tables

For the interface to work correctly the following PIDT and PIPT tables must be in the RFT data set pointed to by the BLGSESxx module.

There are three categories of PIDT/PIPT tables: Change, Change Activity, and Problem.

Change Record PIDT and PIPT Tables

The following PIDT/PIPT tables related to Change records must be installed.

TS0B06A

Change record, Add, PIDT

TS0B06AP

Change record, Add, PIPT

TS0B06I

Change record, Inquiry, PIDT

TS0B06IP

Change record, Inquiry, PIPT

TS0B06C

Change record, Create, PIDT

TS0B06CP

Change record, Create, PIPT

TS0B06U

Change record, Update, PIDT

TS0B06UP

Change record, Update, PIPT

TS0B06R

Change record, Retrieve, PIDT

TS0B06RP

Change record, Retrieve, PIPT

Change Activity PIDT and PIPT Tables

The following PIDT/PIPT tables related to Change Activity records must be installed.

TS0B07I

Change activity record, Inquiry, PIDT.

TS0B07IP

Change activity record, Inquiry, PIPT.

TS0B07C

Change activity record, Create, PIDT.

TS0B07CP

Change activity record, Create, PIPT.

TS0B07U

Change activity record, Update, PIDT.

TS0B07UP

Change activity record, Update, PIPT.

TS0B07R

Change activity record, Retrieve, PIDT.

TS0B07RP

Change activity record, Retrieve, PIPT.

Problem Record PIDT and PIPT Tables

The following PIDT/PIPT tables related to Problem records must be installed.

TS0032I

Problem record, Inquiry, PIDT.

TS0032IP

Problem record, Inquiry, PIPT.

TS0032C

Problem record, Create, PIDT.

TS0032CP

Problem record, Create, PIPT.

TS0032U

Problem record, Update, PIDT.

TS0032UP

Problem record, Update, PIPT.

TS0032R

Problem record, Retrieve, PIDT.

TS0032RP

Problem record, Retrieve, PIPT.