Chicago Interface Group, Inc.

CIG Package Utilities

Reference Guide

Chicago Interface Group, Inc.
858 West Armitage Avenue #286
Chicago, IL 60614 USA

Phone:	(773) 524-0998
Fax:	(815) 550-6088
Internet:	www.cigi.net
Email:	support@cigi.net

CIG Package Utilities 12.0
Copyright © 2008 Chicago Interface Group, Inc. All Rights Reserved
Documentation version January 17, 2008.

CIG PACKAGE UTILITIES
TABLE OF CONTENTS

How to Use This Manual

Breeze and Package Utilities are now combined into a single product offering. The Package Utilities component includes the ISPF end-user interface, batch utilities, and Endevor exits. Breeze includes email notification and the browser interface.

This manual contains information for all TSO/ISPF users. It contains installation, diagnostic, product structure, and usage chapters.

For TSO/ISPF end-users not interested in installation or implementation, focus should be on Concepts and the ISPF end-user interface.

End users who only use the browser front-end and receive email notifications should refer to the Breeze Reference Guide.

Administrators and System Programmers will want to reference both the *Package Utilities Reference Guide* and *Breeze Reference Guide*.

This document assumes that Breeze and CIG Package Utilities have been installed. Refer to the *CIG Installation Guide* for information on how to install CIG's products.

CIG Package Utilities

Chapter 1 Concepts and Facilities

This chapter contains:

- Package Utilities product concept
- Package Utilities usage scenarios
- Related documentation
- Glossary of major terms

Introduction

What are the CIG Package Utilities?

The CIG Package Utilities (Package Utilities) offering is a product designed to help CA-Endevor (Endevor) package users work more efficiently and more effectively. The product is a combination of exits and external utilities that work together to provide an alternate path to Endevor package management. There are several components to the package utility and each will be discussed at length throughout this manual.

The browser and email components of CIG Package Utilities are discussed in the *Breeze Reference Guide*.

Package Utilities Components

- a back-end process installed in Endevor package exits that allows for management of the element collisions, package execution, and reuse of packages
- a utility for printing, archiving, and clearing the audit log
- a utility for reusing an executed package
- a utility for building element and package cross reference reports
- an ISPF front-end for listing against the Package Registry file.

Why use the Package Utilities if you have Endevor?

Package Utilities enhances and simplifies Endevor package processing. You can now reuse an existing package, reducing package build time, administrative overhead, and DASD usage. Packages can be rebuilt automatically, promoted to the next stage, or just regenerated in place. You have only one central place to look for audit trail information, listing the “who, what and when” for each package. Finally, with four different options, you can decide what action to take if an element collision occurs on a **CAST** action or with a non-package action. The following section of this chapter discusses each feature through some typical Endevor package usage scenarios.

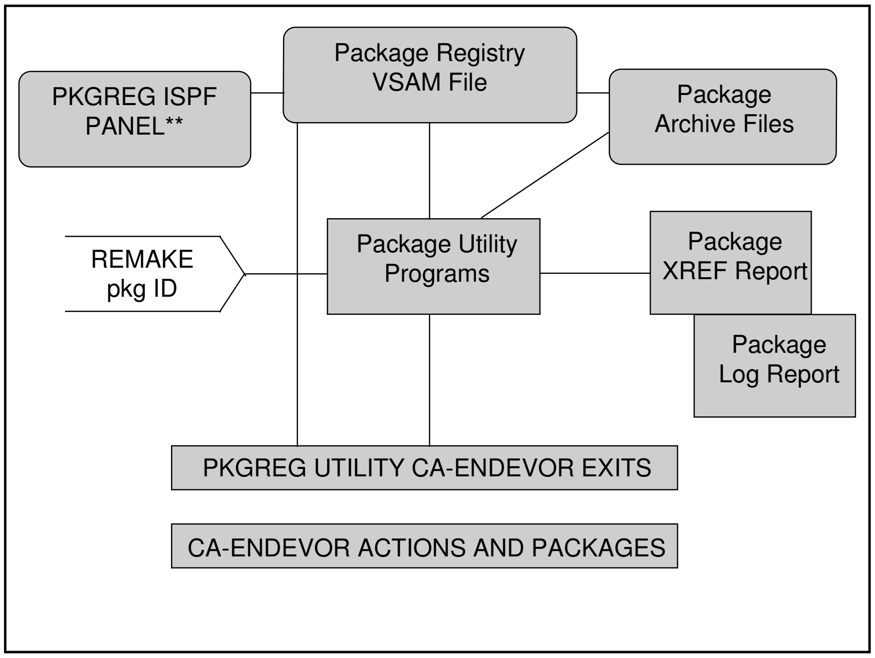


Figure 1.1
Package Utilities Components

Figure 1.1 shows the components of the Package Utilities. There are two external utilities, CIGPKUT1 and CIGPKUT2; two reports, the audit trail report and cross-reference report; the exits; and an ISPF front end for report generation and file listing.

What happens to existing, 'in progress' packages?

The Package Utility will begin working with packages at any stage of the package life-cycle. The first time any package is acted upon, after the package exits are installed, the Utility will determine if the package has ever been registered in the Package Registry File. If the answer is no, then the package is registered at that time. For example, if you have a dozen packages that are 'in-edit' status, then the Utility would intercept these packages as soon as the package is **MODIFIED** or **CAST**. The only exception is packages that have been **CAST**, but not **APPROVED**. Those packages should be **RESET** and **CAST** again, to obtain element registration.

If the package is already approved, the product would intercept the package during the '**before exec'** exit point, etc.

Is there a way to transfer existing, historical packages?

If you want to offload all historical packages, there is an ARCHIVE exit point that will perform a one-time extraction of the package data and build a log for the data. Minimally, each package will have log data that reflect the major dates and users of the package. In addition, approvers and action data will also be transferred to the log, as if the log were active at the time of approval and execution.

To invoke the transfer process, you must execute the Endevor **PACKAGE ARCHIVE** command against the packages you wish to have offloaded. There is no additional interface to learn or invoke. The Utilities will automatically transfer the data from the ‘before archive’ exit point. Once this data is transferred to the Package Utilities Registry, the user can then perform an Endevor **PACKAGE DELETE** command and clean all the ‘historical’ data out of the package file.

Usage Scenarios - Reusable Packages

The Need for Reusable Packages

The most common use of the Endevor package facility is to promote a predefined list of elements via a package up the life cycle map. Typically, programmers or managers will build a package at the entry point of the map and then request that the elements in the list be promoted up the map in a series of moves. The number of packages created is determined by the number of stages in the life cycle.

Enter the Package Utilities. After a package is created in Endevor, the utility monitors every action against the package ID. Also, after the package has been successfully executed, it is eligible to be reused. Either automatically or via the use of a batch utility, the old package will be remade, the element list will be promoted and batch package SCL will be created to recycle the package ID in Endevor. All of this happens invisibly and without losing the previous history of the package.

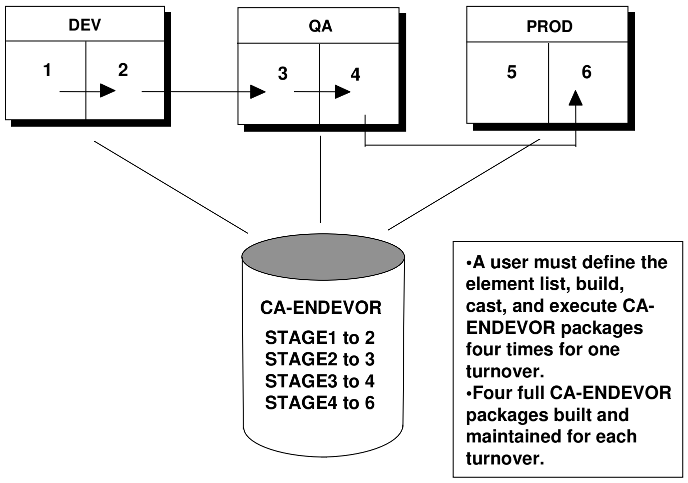

DEV
1
2
QA
3
4
PROD
5
6
CA-ENDEVOR
STAGE1 to 2
STAGE2 to 3
STAGE3 to 4
STAGE4 to 6
•A user must define the element list, build, cast, and execute CA-ENDEVOR packages four times for one turnover.
•Four full CA-ENDEVOR packages built and maintained for each turnover.

Figure 1.2
An Example of One Turnover without Package Utilities

Figure 1.2 shows a typical Endevor map. Without Package Utilities, a programmer or administrator must rebuild the package from a fresh element list at every stage. Because of audit trail requirements, the package is often built with a new name to retain the old package information.

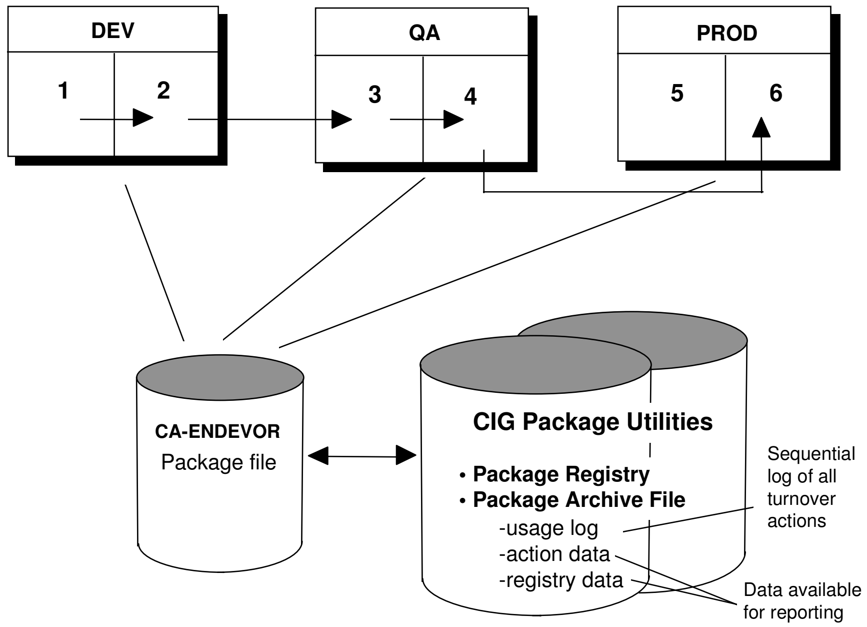


Figure 1.3
Single Package per Turnover

Figure 1.3 shows a typical Endevor map with the Package Utilities. Through the use of the Package Utilities REMAKE facility, the programmer only has to build the element list once and the list will be moved up the map automatically. This enhancement allows for one package ID per turnover, while still retaining a complete, auditable history of the turnover activity.

For more information on how to REMAKE a package, see chapter 5, *Reusable Packages,* which details how to invoke a REMAKE and how to control the REMAKE process via user options and a user exit point for further customization.

Usage Scenarios - Element Collision Management

Element Collisions Due to Parallel Development

In the best of all worlds, a single programmer would work on a single piece of code at one time. All updating would occur at a controlled pace and would be sequential in nature. In contrast, most Endevor users are faced with ever increasing demands for systems changes, both long term and short term. In many shops, the same module may be updated by different people, teams, or locations. This parallel development activity can lead to code regressions and collisions if not managed. With Endevor packages, an element is CAST into a package but not frozen from use by other packages or stand-alone actions. Even after being CAST, the element can be retrieved, updated, deleted, CAST into another package, or even moved by another package, causing package failures and incorrect code implementations.

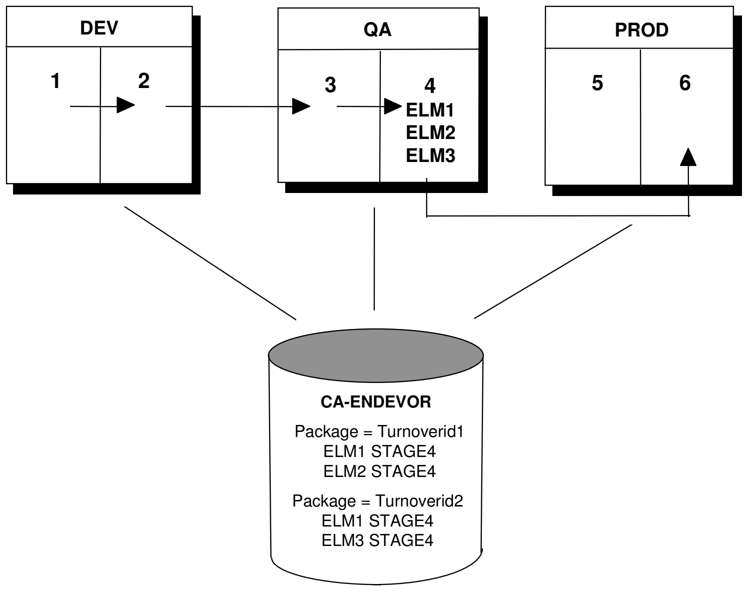


Figure 1.4
Same Element Attached to Multiple Packages

Figure 1.4 shows a case where ELM1 has been CAST into two different packages. In the event ELM1 is subsequently modified, package failure will occur and the incorrect level of element ELM1 will be promoted.

Enter the Package Utilities. At Endevor **CAST** time, each element is both checked against an element registry of existing packages and registered to the current package if the package passes the collision tests. If the element is found to be registered to another package ID, then one of four events occur, based on CIGINI options:

- the package is failed,
- the package is set to "resolve" with all the other packages involved in the collisions,
- the user is warned of a collision, or
- the collision is ignored.

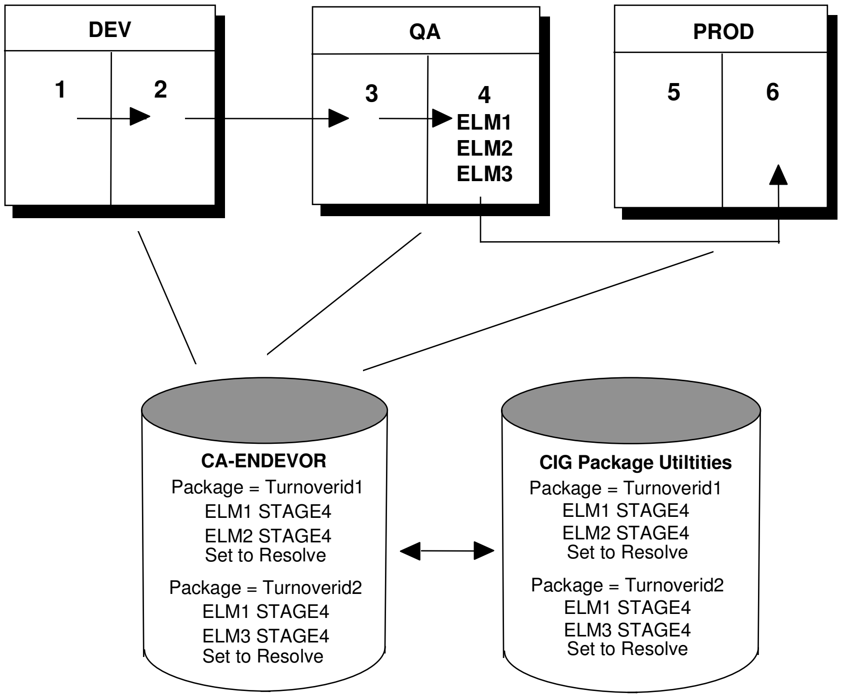


Figure 1.5
CAST Collision Checks and Events

Figure 1.5 shows how the Element Registry records are used to help you manage and correct element collisions. In this case, Package Turnoverid1 was ***CAST*** *first and its elements were registered. When Turnoverid2 was* ***CAST****, there was an element collision with Turnoverid1. The package was fully* ***CAST****, but set to RESOLVE status. Turnoverid1 was also set to RESOLVE status. Prior to either of these packages being executed or approved, they would have to be* ***RESET*** *(or*

RESETID*) and the collision would need to be resolved. The log of each package ID would be updated to reflect the collisions, warnings, and resolution.*

Note that elements remain "registered" to package from CAST time until 1) the package is RESET, 2) the package is EXECUTED, or 3) the package is DELETED.

Avoiding Production Overlays

Another aspect of collision management is detecting and preventing element overlays during move and transfer processing. All Transfer action targets that are Endevor locations will be registered at CAST time. Therefore, not only is the source monitored, but also the target. Since the destination of the Move action is not determined until execution time, the target of a Move action will be checked for collisions during exit 2. If the target of the Move action is registered to another package, the move action will be canceled or a warning will be issued, as per the CIGINI settings for collisions. This is true for Move actions inside or outside of a package.

User Notification Tools

Whenever a collision occurs during CAST, the creator and last update userid will be notified using the Endevor notification facility. The severity of the collision will be reflected in the message. You can also opt to include an optional user exit that will notify all approvers of the current package of the collision. The source for this optional exit is also provided for further user customization.

For more information on managing collisions with the Package Utility, see *Step 2: Review Options and Build the CIGINI Module* in chapter 3, *Implementation*, or *Managing Element Collisions via Exits* in chapter 4, *Package Exits and Collision Management*.

Usage Scenarios - Audit Trails & DASD Usage

Full History of Package ID Activity

Most Endevor users have very serious auditing requirements that require them to keep an audit trail of all element activity. Traditional Endevor package processing builds many records per package ID. If the package ID is used again, then all history is lost, and thus the audit trail is lost. Most shops end up keeping fully executed package ids for many months for audit reasons. This causes the package file to get huge, radically affecting performance and DASD costs.

Enter the Package Utility. Every time the package ID is updated, a log record gets cut. These records do not go away when the package is RESET or even DELETED out of Endevor. The package may be CAST and RESET several times, and the audit trail will contain the information. Therefore, if you CAST then RESET and MODIFY the package, each of those actions will be recorded. All package activity is recorded and historically maintained, even if the package ID is re-used. This audit trail data can also be extracted for additional reporting and auditing activities.

Figure 1.6 shows an example of a log for a package that has been built, executed, and processed via the REMAKE process. In this example, two elements have been moved twice from stage A then from stage B.

```
DATE 08/01/13 TIME 16:59:54     P A C K A G E   U T I L I T Y, RELEASE 12.0      PAGE  1
                                  A U D I T  L O G   R E P O R T
FOR PACKAGE: TEST1    STATUS: APPROVED      UTILSTAT:      UPDATE: 08/01/13  13:07 USER:

      ACTIVITY        USER        DATE       TIME     RC

      CREATE          CIG01A      08/01/13   12:57    00
      CAST            CIG01A      08/01/13   12:57    00
        MOVE        $CPOOL      (01.00)   TEST       SYSA        SUBA         MAC  A
        MOVE        $ENTRY      (01.00)   TEST       SYSA        SUBA         MAC  A
      EXECUTE         CIG01A      08/01/13   13:01    04
      REMAKE          TPSXXX      08/01/13   13:01    00
      COMMIT          TPSXXX      08/01/13   13:02    00
      DELETE          TPSXXX      08/01/13   13:02    00
      CREATE          TPSXXX      08/01/13   13:02    00
      CAST            TPSXXX      08/01/13   13:02    00
        MOVE        $CPOOL      (01.00)   TEST       SYSA        SUBA         MAC  B
        MOVE        $ENTRY      (01.00)   TEST       SYSA        SUBA         MAC  B
      EXECUTE         CIG01A      08/01/13   13:06    00
      REMAKE          CIG02A      08/01/13   13:06    00
 16:59:54  PKG3163I  PRINTLOG COMPLETED SUCCESSFULLY FOR PACKAGE 'TEST1' .
```

Figure 1.6
Audit Trail for Package 'Test1'

Immediate Offload of History and Reduction of Package File Size

You may want to transfer historical package ids to the Package Utilities Registry File. This transfer would allow Endevor Administrators to **DELETE** the historical packages from the Endevor package file, thus limiting the file to only active packages.

To offload the historical packages from the Endevor package file, you will need to execute the Endevor **PACKAGE ARCHIVE** action against the segment of packages to be offloaded. This action will cause the Package Utility Exits to detect that a non-registered package is being archived. The Utility will then register the package, build action records, and build log records that reflect the creator, caster, approvers, and actions executed for the package. This will be a one-time shot to capture all key audit information about the package.

After you successfully perform an Endevor **ARCHIVE** of the package files, you should perform an Endevor **PACKAGE DELETE** action to clear out the historical packages from Endevor. The net result is a radical reduction of size of the package file. Note that while you can use the Endevor **ARCHIVE** with the **DELETE** option, it is recommended that each step be performed to ensure that the required package information is offloaded.

Glossary of Common Terms

The following is a list of terms used throughout this manual.

Package Registry File	This is a VSAM control file used by the utility to maintain all header, log, and action records.
CIGPKUT1 and CIGPKUT2	Utility programs that perform various reporting and actions against the Package Registry File.
REMAKE	The command verb for recycling a package.
Utility Status	A status field maintained by the Package Utility.
Registered element	An element that is CAST into a package, but not executed.
Registered package	A package that is being managed and

	monitored by the Package Utility.
CIGINI	A C1DEFLTS like setup module for installing user options and passwords.
CIGINI override	A secondary CIGINI module for pointing to a different Registry File.
Action Summary Records	Package Utility Records for maintaining action/element information, per package.

Chapter 2 Implementation

This chapter contains a step-by-step guide to implementing the Package Utilities, including instructions on:

- Reviewing all the user CIGINI options
- Build site CIGINI module
- Modify file skeletons
- Enabling the Package Utilities Exits
- Loading Historical Package Data from Endevor

Implementation

The Package Utilities is a set of exits and utilities designed to work inside of and around Endevor. The exits act as control and collection agents, and the outside utilities provide reporting and package REMAKE options. The product is implemented as soon as the exits are installed and the password is turned on via the CIGINI file.

It is very important that you review all the CIGINI options for package management and consider using FILTERS, at least for the first phase of implementation. We recommend that for the initial phase collision options be set to 'WARN'. This will give both the implementation team and the users a chance to see how much colliding is going on. After the users have been trained on what to do in the case of a collision, turn the collision option onto RESOLVE or FAIL.

To implement and enable the Package Utilities, follow the following steps:

STEP	ACTION
❶	Fill out the Package Utilities worksheet.
❷	Review options and build the CIGINI module.
❸	Allocate the Package Registry file.
❹.	Modify file skeletons.
❺	Modify Endevor file skeletons.
❻	Modify REMAKE shell.
❼	Enable the EXITS.
❽	Optionally, offload old audit information.
❾	Optionally, install a REMAKE Exit program.

Figure 3.1
Implementation Step-by-step

Step 1: Fill out the Package Utilities Worksheet

The purpose of this worksheet is to help the user make decisions about what options to implement in the CIGINI module. Answer the following questions about your current and desired package life cycle and process. Please refer to the Breeze for Endevor User Guide and Package Utilities Functional Overview for a high level review of most of these concepts. This reference guide contains detailed information on element collision and remake. In the beginning, you may not have the answers to the questions; but after you have some familiarity with the product and its usage, this worksheet will be a useful tool.

1. Do you want to automatically rebuild a package after a successful execution?
2. Do you want to invoke an exit anytime that a package is rebuilt from the REMAKE facility?
3. If a package contains elements that are already cast into another package, which of the following actions do you want installed?
   - Fail the current package with notification
   - Set the current package and all others with collisions into a RESOLVE or HOLD status
   - Send a warning to current packages and colliding packages that an element collision has occurred.
   - Ignore element collisions.
4. Do you want to use the audit trail facility?
5. Do you use EMERGENCY packages and would they qualify for the REMAKE process?
6. Do you employ a naming standard for packages?
7. Are there some packages that you want monitored and others not?
8. Are you using Endevor approver groups?
9. Do you use Endevor actions other than Move your packages?

Step 2: Review Options and Build the CIGINI Module

The CIGINI file is an initialization module that contains the following Package Utilities parameters:

1. The name of the Package Registry File
2. The name of the Package Utilities LOADLIB
3. Package Utilities Options
4. Package Utilities FILTERS
5. Product password
6. VIO and Work units
7. Other administrative options

Every site must build a primary CIGINI module for installation in a steplib or job pack area. The CIGINI module is very similar to the Endevor C1DEFLTS module, in that both are loaded at initialization and values will be used throughout the use of the application. The CIGINI module is always used for the password check, and contains the default initialization parameters for the Package Utilities. Even if you determine that you want more than one Package Utilities Registry File, you must still create a primary CIGINI module with a default database. Additional modules and databases can be added after the creation of this first module.

Building the CIGINI Module

During install, you created an initialization module. In this step, you will rebuild this module to reflect the options and database desired by your site. This will be your primary CIGINI module, and will serve as the default setting for all initialization parameters. During install, you also modified member *flhq1.flhq2*.SAMPLIB (CIGINI) to include your installations naming standards and to set up the test database. Figure 3.0 on the next page shows this syntax. You should access this member in ISPF edit and change the variables to reflect your desired options for your primary CIGINI module.

Parameters listed below pertain to the 'common section', the 'FASTLIST SECTION', and the 'PACKAGE SECTION'. If you are installing only the Package Utility, delete the 'FASTLIST SECTION'. The remainder of this chapter reviews only those parameters that pertain to the Package Utilities. Refer to the FastLIST Administration Guide for a description of the FastLIST CIGINI parameters.

Sample CIGINI Source

```
DEFINE COMMON SECTION
PRODUCT LOADLIB     = ‘flhq1.flhq2.LOADLIB’
WORK UNIT           = WORK
VIO  UNIT           = WORK
DO NOT ALLOW ALTERNATE CIGINI FILE
ENDEVOR CONLIB DSNAME = ‘qual1.conlib’
JAVASERVERCONTROL DSNAME = ‘breeze.JAVALIB’
NOTIFY RULES DSNAME = ‘breeze.JAVALIB’

DEFINE FASTLIST SECTION
FASTLIST PASSWORD = ‘password’
VSAM DSNAME = ‘flhq1.fastlist.database’

DEFINE BREEZE SECTION
PASSWORD = ‘password’

DEFINE PACKAGE SECTION
PACKAGE PASSWORD =‘password’
RESOLVE
*WARN | FAIL | RESOLVE | IGNORE ELEMENT COLLISIONS
PACKAGE LOG IS REQUIRED
PACKAGE VSAM DSNAME = ‘flhq1.PACKAGE.DB’
AUTOMATIC REMAKE
USER PROGRAM BEFORE REMAKE = ‘TESTPGM’
DO NOT REMAKE EMERGENCY PACKAGES
DISABLE DELETE
ACTION OPTIONS FOR
MOVE = MOVENEXT
GENERATE = ASIS
TRANSFER = ASIS
ADD = ASIS
UPDATE = ASIS
RETRIEVE = ASIS
DELETE = ASIS
PRINT = ASIS
LIST = ASIS
ARCHIVE = ASIS
RESTORE = ASIS
*** ASIS DISCARD (MOVENEXT for MOVE action only)
```

Figure 3.2
Sample CIGINI Source

CIGINI Syntax Review

You should have modified the Package Utilities LOADLIB and Password to reflect your site-specific variables during install. Use figure 3.3 to determine your desired parameters, and change the above syntax accordingly.

SYNTAX	Purpose	Default
DEFINE COMMON SECTION	SIGNIFY BEGINNING OF COMMON SECTION	REQUIRED.
PRODUCT LOADLIB =	Loadlib containing The Package Utilities.	None.
WORK UNIT =	OPTIONAL. Defines DASD unit name for temporary disk files.	SYSDA
VIO UNIT =	OPTIONAL. Defines VIO name for VIO usage.	VIO
ENDEVOR CONLIB DSN	REQUIRED. The name of the data set containing Endevor CONLIB modules.	None.
JAVASERVER CONTROL DSNAME	REQUIRED for Breeze.	None
NOTIFY RULES DSNAME	REQUIRED for Breeze.	None
[DO NOT] ALLOW ALTERNATE CIGINI	OPTIONAL. Prevents use of override/alternate initialization modules.	Override init use allowed
DEFINE PACKAGE SECTION	Signify Beginning of Package Section.	Required for Package Utility.
PACKAGE PASSWORD	REQUIRED. User will be provided this password at install time.	None

RESOLVE I WARN I FAIL I IGNORE ELEMENT COLLISIONS	User option for determine event at element collision. Note RESOLVE applies only to CAST. If non-package action collision occurs, Resolve will be converted to Fail Action.	Fail.
PACKAGE LOG IS [NOT] Required	Determines if Utilities will update log.	LOG
PACKAGE VSAM DATABASE =	REQUIRED. The name of the VSAM Package Registry File.	None.
AUTOMATIC REMAKE	OPTIONAL. Automatically REMAKE successfully executed packages.	No automatic REMAKE.
User Program For Remake =	OPTIONAL. The name of a user supplied program to call prior to REMAKE. User will be passed a $USRDS block to fill in. If the user wants to cancel the REMAKE, then they should issue a non-zero return code in Register 15. See Special Remake Considerations section.	The default is no-program. Package will be rebuilt using the attributes and name of the current package ID.
DO NOT REMAKE EMERGENCY PACKAGES	OPTIONAL. Only REMAKE 'standard' Endevor packages	All packages qualify.
DISABLE DELETE	OPTIONAL. Disables the use of the Package Utilities DELETE and CLEARLOG functions.	All packages qualify.
FILTER (pkgid*, pkgid*)	OPTIONAL. Limit Package Utilities to packages that meet these naming standards.	All packages qualify.

MOVE	ASIS | DISCARD | MOVENEXT	ASIS means attempt to re-execute the action the way it is. DISCARD means to not include in the newly rebuild package. MOVENEXT means to build a move statement using the target of the MOVE action summary.
GENERATE	ASIS | DISCARD	
TRANSFER	ASIS | DISCARD	
ADD | UPDATE	ASIS | DISCARD	
RETRIEVE	ASIS | DISCARD	
DELETE	ASIS | DISCARD	
PRINT	ASIS | DISCARD	
LIST	ASIS | DISCARD	
ARCHIVE	ASIS | DISCARD	
RESTORE	ASIS | DISCARD	

Figure 3.3
Session and Action Options

Once the CIGINI syntax has been modified to reflect your preferences and Endevor package filters, submit CIGJCL04. This JCL should already have been modified to include your site-specific variables during install. Point the SYSLMOD statement directly to your Endevor authorized library or whatever STEPLIB data set you are using for the Package Utilities “Required Modules”. Otherwise, make sure that you copy the CIGINI file into that STEPLIB or LINKLIST.

You have now set up your primary CIGINI module and the Package Utilities applications are ready to be executed. Note that all the Package Utilities applications will use the database and options from this primary CIGINI module.

Step 3: Allocate the Package Utilities Registry File

Size the JCL

Prior to creating your Package Utilities Registry File, you should size the JCL to meet your requirements, especially if you are going into production. The Package Utilities Registry Control File has one copy of the current package and a historical log that can be printed, archived, and deleted as per the users request. The JCL delivered with the product should suffice for initial implementation, but you should monitor the size of the file for expansion. If you are planning on using more than one database/CIGINI for your full implementation, you will need to complete this step for each database required.

Remember to review the user and security requirements before determining the data set name.

Allocate the Package Registry File

During install, you used CIGJCL50 to allocate the sample database. To build your pilot Registry File, modify CIGJCL55 to build your production file. Note that there is an additional step in this JCL stream. Step 1 will create a small data set used to initialize the empty VSAM file.

```
//**(JOBCARD)
//**
//* ---------------------------------------------------------------- *
//* NAME:    CIGJCL55                                                *
//* PURPOSE: ALLOCATE THE PACKAGE REGISTRY FILE                      *
//* ---------------------------------------------------------------- *
//* -----------------------------------------------------------------*
//*  TO USE THIS JCL, YOU MUST:                                      *
//*         1) INSERT A VALID JOB CARD WITH VALID CLASS.             *
//*         2) CHANGE FLHQ1, FLHQ2, QUAL1 AND QUAL2 AS PER YOUR      *
//*            INSTALLATION SHEET                                    *
//*         3) CHANGE THE UNIT=WORK  TO THE APPROPRIATE WORK UNIT    *
//*            NAME.                                                 *
//*         4) CHANGE THE DVOLSER VALUE AS PER WORKSHEET. IF YOUR    *
//*            SITE IS USING SMS FOR VSAM FILE ALLOCATION, THEN      *
//*            YOU MAY NOT NEED THIS PARAMETER.                      *
//* -----------------------------------------------------------------*
//* ---------------------------------------------------------------- *
//* ALLOCATE A TEMPORARY INITIALIZATION FILE                         *
//* ---------------------------------------------------------------- *
//STEP1    EXEC PGM=CIGFPOPL
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGOUT   DD  DSN=&&TEMP,DISP=(NEW,PASS),
//             SPACE=(1,1),UNIT=WORK,
//             DCB=(BLKSIZE=187,LRECL=187,RECFM=FB)
//CIGLOG   DD  SYSOUT=*
```

```
//* -------------------------------------------------------------------- *
//* ALLOCATE THE PACKAGE REGISTRY CONTROL FILE                           *
//* -------------------------------------------------------------------- *
//STEP2   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INDD01   DD  DSN=&&TEMP,DISP=(OLD,DELETE)
//SYSIN   DD *
DELETE FLHQ1.PACKUTIL.DB
DEFINE CLUSTER -
(NAME('FLHQ1.PACKUTIL.DB') -
SPEED UNIQUE FREESPACE(30 30) -
CYLINDERS(1  1 ) -
VOLUMES(DVOLSER) -
SHR(3 3) -
KEYS(80 0)  -
CISZ(16000) -
RECORDSIZE(80 600))
REPRO INFILE(INDD01) OUTDATASET('FLHQ1.PACKUTIL.DB')
```

Figure 3.4
CIGJCL55

Maintain the Package Utilities Registry File

The Package Utilities Registry File is a KSDS VSAM file. To maintain the file, it is recommended that you run a reorganization job and rebuild the database on a regular basis. Member CIGJCL53 in the JCL library contains JCL to perform a sequential backup of the database, delete and define the database, and repro the sequential backup into the new VSAM file. Prior to running this job verify that the define parameters reflect the parameters defined for CIGJCL55.

Step 4: Modify File Skeletons

There are two file tailoring skeletons that get delivered with the product, CIGSKL01 and CIGSKL02. These skeletons can be found in the 'ISPSLIB' data set offloaded from the tape. Modify these skeletons as per instructions. Be sure to delete various comments (<= some comment) in the skeleton.

Modify CIGSKL01

The following skeleton will be expanded and submitted via the Package Utilities ISPF front end. When you enter a REPORT, REPORTX, PRINTLOG, or RESETID request, the CIGSKL01 file skeleton will be filled in with job card information and input parameters. At this time, modify the STEPLIB and CIGARCH data set names. Note that this skeleton is shared amongst all users. All variables (fields that start with &) will be filled in at execution time from the users' ISPF profile variables. These values are not shared between users.

```
)CM  THIS SKELETON IS USED TO GENERATE PACKAGE UTILITY JCL FOR BATCH.
)SEL &C1BJC1 ^= &Z
&C1BJC1
)ENDSEL
)SEL &C1BJC2 ^= &Z
&C1BJC2
)ENDSEL
)SEL &C1BJC3 ^= &Z
&C1BJC3
)ENDSEL
)SEL &C1BJC4 ^= &Z
&C1BJC4
)ENDSEL
//*
//* ------------------------------------------------------------------*
//* NAME:    CIGSKL01                                                 *
//* PURPOSE: FILE  SKELETON FOR BUILDING CIGPKUT1 JCL                 *
//* ------------------------------------------------------------------*
//* ------------------------------------------------------------------*
//*  MODIFY ALL DATASETS TO MEET SITE REQUIREMENTS                    *
//* ------------------------------------------------------------------*
//STEP1    EXEC PGM=CIGPKUT1
//STEPLIB  DD  DSN=flhq1.flhq2.LOADLIB,DISP=SHR
//CIGLOG   DD  SYSOUT=*
//CIGRPT   DD  SYSOUT=*
//* ONLY FOR ARCHLOG
//CIGARCH  DD  DSN=flhq1.ARCHDATA,DISP=SHR
//* ------------------------------------------------------------------*
//*  INPUT DATASET   ( SHARED WITH ENDEVOR AND FASTLIST)              *
//* ------------------------------------------------------------------*
//CIGIN    DD DSN=&VNBDFDSN,DISP=SHR
```

Figure 3.5
CIGSKL01

Modify CIGSKL02

The following skeleton will be expanded and submitted via the Package Utilities ISPF front end. When you enter a REMAKE or DELETE request, the CIGSKL02 file skeleton will be filled in with job card information and input parameters. At this time, modify all data sets names and unit values for your configuration. Note that step 2 of this skeleton is an Endevor batch package execution. This skeleton is shared amongst all users. All variables (fields that start with &) will be filled in at execution time from the users' ISPF profile variables. These values are not shared between users.

```
)CM  THIS SKELETON IS USED TO GENERATE PACKAGE UTILITY JCL FOR BATCH.
)SEL &C1BJC1 ¬= &Z
&C1BJC1
)ENDSEL
)SEL &C1BJC2 ¬= &Z
&C1BJC2
)ENDSEL
)SEL &C1BJC3 ¬= &Z
&C1BJC3
)ENDSEL
)SEL &C1BJC4 ¬= &Z
&C1BJC4
)ENDSEL
//* ---------------------------------------------------------------------*
//* REQUIRED SKELETON MODIFICATION:                                      *
//*   1) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.         *
//*      - FLHQ1 AND FLHQ2                                               *
//*      - QUAL1 AND QUAL2                                               *
//*      - TDISK                                                         *
//*                                                                      *
//* THE FOLLOWING MAY NOT REQUIRED FOR SMS INSTALLATIONS:                *
//*   - TDISK                                                            *
//* -------------------------------------------------------------------- *
//* ---------------------------------------------------------------------*
//* NAME:    CIGSKL02                                                    *
//* PURPOSE: FILE TAILORING SKELETION FOR BUILD CIGPKUT2 JCL FROM THE *
//*          ISPF FRONT-END.                                             *
//* ---------------------------------------------------------------------*
//* --------------------------------------------------------------- *
//* JCL TO RUN THE REMAKE AND DELETE PACKAGE UTILITY COMMANDS.      *
//* IF NO ENDEVOR SCL WAS WRITTEN THEN THE RETURN CODE WILL BE 4.   *
//* --------------------------------------------------------------- *
//STEP1    EXEC PGM=CIGPKUT2
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGSCL01 DD  DSN=&&TEMP1,DISP=(NEW,PASS),
//             UNIT=TDISK,SPACE=(CYL,(1,1)),
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGSCL02 DD  DSN=&&TEMP2,DISP=(NEW,PASS),
//             UNIT=TDISK,SPACE=(CYL,(1,1)),
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGLOG   DD  SYSOUT=*
//* -------------------------------------------------------------------- *
//*  BUILD INLINE INPUT STREAM.                                          *
//* -------------------------------------------------------------------- *
)SEL &VACTION = DELETE
//CIGIN    DD *
 DELETE PACKAGE &VLPKGID .
/*
)ENDSEL
)SEL &VACTION = REMAKE
//CIGIN    DD *
```

```
 REMAKE PACKAGE &VLPKGID .
/*
//* UNCOMMENT TO ACTIVATE SYNTAX GENERATION OVERRIDE
//* //$BOENA   DD  DUMMY     BACKOUT ENABLED
//* //$CMPVALW DD  DUMMY     VALIDATE COMPONENTS WITH WARNING
//* //$CMPVALY DD  DUMMY     VALIDATE COMPONENTS
//* //$CMPVALN DD  DUMMY     DO NOT VALIDATE COMPONENTS
//* //$SHRABLE DD  DUMMY     SHARABLE PACKAGE
)ENDSEL
//* ---------------------------------------------------------------- *
//* THIS JCL WILL EXECUTE  ENDEVOR PACKAGE SCL BUILT IN STEP ONE.    *
//* ---------------------------------------------------------------- *
//IFSTEP1 IF STEP1.RC = 0 THEN
//STEP2   EXEC PGM=NDVRC1,PARM='ENBP1000'
//STEPLIB DD DSN=QUAL1.QUAL2.LOADLIB,DISP=SHR
//CONLIB  DD DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//C1MSGS1  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//CIGSCL01 DD  DSN=&&TEMP1,DISP=(OLD,DELETE,DELETE)
//ENPSCLIN DD  DSN=&&TEMP2,DISP=(OLD,DELETE,DELETE)
//* ---------------------------------------------------------------- *
//* PACKAGE UTILITY DATASETS                                         *
//* NOTE THE TRACE IS OPTIONAL                                       *
//* ---------------------------------------------------------------- *
//CIGOUT   DD  SYSOUT=*
//* CIGTRACE DD  DUMMY                UNCOMMENT IF TRACING DESIRED
)SEL &VNBINCF = Y
//* ------------------------------------------------------------------ *
//*  ADDITIONAL JCL STATEMENTS (SHARED WITH ENDEVOR AND FASTLIST)      *
```

Figure 3.6
CIGSKL02 File Skeleton

Step 5: Modify Endevor File Skeletons

Modify C1SB3000

The standard batch interface to Endevor uses the C1SB3000 file skeleton to submit both stand-alone batch actions as well as the submit option from the foreground package menu. The new batch package interface uses the ENSP1000 file skeleton. Check with your Endevor administrator to see if these skeletons have been modified already. Add the following ddname and comments to both Endevor skeletons.

```
//*
//* THE CIG MESSAGE OUTPUT DATASET
//*
//CIGOUT    DD   SYSOUT=*
```

If AUTOMATIC REMAKE is going to be in effect, add the following ddnames and comments to both skeletons.

```
//* The JES2 INTERNAL READER DD.  This file will be used to
//* pass the modified JCL to JES2.
//CIGINRDR DD       SYSOUT=(A,INTRDR)

//* The external user JCL DD. This file should contain the
//* REMAKE shell from the JCLLIB library,  member called
//* CIGJCL56.  Users should modify this JCL to include a JOB
//* card.
//*
//CIGJCLPK DD      DSN=flhq1.samplib(cigjcl56),DISP=SHR
```

Step 6: Modify REMAKE Shell

If you code the AUTOMATIC REMAKE option in the CIGINI file, then the system will automatically spawn a batch job to REMAKE the current package. To facilitate this automatic process you will need to include two additional JCL cards in your Endevor batch package JCL and your standard Endevor batch skeletons ENSP1000 and C1SB3000 (see previous step). You will also need to customize the REMAKE JCL shell provided in the SAMPLIB, member name CIGJCL56.

Modify CIGJCL56

```
//**(JOBCARD)
//**
//* -----------------------------------------------------------------*
//* NAME:    CIGJCL56                                                *
//* PURPOSE: EXAMPLE JCL FOR INVOKING PACKAGE REMAKE                  *
//* -----------------------------------------------------------------*
//* -----------------------------------------------------------------*
//* REQUIRED JCL MODIFICATION:                                       *
//*   1) INCLUDE A JOBCARD                                           *
//*   2) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.     *
//*      - FLHQ1 AND FLHQ2                                           *
//*      - QUAL1 AND QUAL2                                           *
//*      - UNIT=TDISK                                                *
//*                                                                  *
//* -----------------------------------------------------------------*
//* FOR PACKAGE WARN MODE UNCOMMENT //C$REMAKE DD DUMMY     Z021213C *
//*                                                                  *
//*   ADD DD STATEMENTS TO ADJUST GENERATED SYNTAX:         Z130709A
//*          //$BOENA   DD  DUMMY     BACKOUT ENABLED
//*          //$CMPVALW DD  DUMMY     VALIDATE COMPONENTS WITH WARNING
//*          //$CMPVALY DD  DUMMY     VALIDATE COMPONENTS
//*          //$CMPVALN DD  DUMMY     DO NOT VALIDATE COMPONENTS
//*          //$SHRABLE DD  DUMMY     SHARABLE PACKAGE      Z140713A
//*   DEFAULT WILL BE:
//*                                 BACKOUT NOT ENABLED
//*                 (7.0 or higher)  INHERIT VALIDATION OF PREV CAST
//*                 (pre 7.0)        SITE DEFAULT BASED ON C1DEFLTS
//* -----------------------------------------------------------------*
//* JCL TO RUN THE REMAKE AND DELETE PACKAGE UTILITY COMMANDS.       *
//* IF NO ENDEVOR SCL WAS WRITTEN THEN THE RETURN CODE WILL BE 4.    *
//* -----------------------------------------------------------------*
//STEP1    EXEC PGM=CIGPKUT2
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGSCL01 DD  DSN=&&TEMP1,DISP=(NEW,PASS),
//             UNIT=TDISK,SPACE=(1,1),
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGSCL02 DD  DSN=&&TEMP2,DISP=(NEW,PASS),
//             UNIT=TDISK,SPACE=(1,1),
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGLOG   DD  SYSOUT=*
//* UNCOMMENT TO ACTIVATE SYNTAX GENERATION OVERRIDE
//*$BOENA   DD  DUMMY     BACKOUT ENABLED
//*$CMPVALW DD  DUMMY     VALIDATE COMPONENTS WITH WARNING
//*$CMPVALY DD  DUMMY     VALIDATE COMPONENTS
//*$CMPVALN DD  DUMMY     DO NOT VALIDATE COMPONENTS
//*$CMPVALN DD  DUMMY     DO NOT VALIDATE COMPONENTS
//*$SHRABLE DD  DUMMY     SHARABLE PACKAGE
```

```
//CIGIN    DD  *
** INCLUDE REMAKE COMMAND SYNTAX HERE **
/*
//* -------------------------------------------------------------- *
//* THIS JCL WILL EXECUTE  ENDEVOR PACKAGE SCL BUILT IN STEP ONE.   *
//* FOR PACKAGE WARN MODE UNCOMMENT //C$REMAKE DD DUMMY            *
//* -------------------------------------------------------------- *
//IFSTEP1 IF STEP1.RC = 0 THEN
//STEP2   EXEC PGM=NDVRC1,PARM='ENBP1000'
//STEPLIB  DD  DSN=QUAL1.QUAL2.LOADLIB,DISP=SHR
//CONLIB   DD  DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//C1MSGS1  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//CIGSCL01 DD  DSN=&&TEMP1,DISP=(OLD,DELETE,DELETE)
//ENPSCLIN DD  DSN=&&TEMP2,DISP=(OLD,DELETE,DELETE)
//*C$REMAKE DD  DUMMY
//CIGOUT   DD  SYSOUT=*
//      ENDIF
```

Figure 3.7
Modify CIGJCL56

Step 7: Enable the EXITS

The Package Utility Exits are the key to the Package Utility. These exits provide a rule set and collection mechanism for streamlining package and collision management. Remember - once you have updated the Endevor exit table, the product is installed. It is highly recommended that you read chapter 4, *The Package Utilities Exits*, prior to enabling the exit interface.

To enable the exits, add the Package Utilities modules into your current CIUEXITS table input and reassemble. The Package Utilities modules required per exit point are listed in the box below. If your current Endevor implementation includes user exits, review with the Endevor administrator how to include the Package Utilities exits. It is recommended that the Package Utilities exit module be placed first in the user exit list of C1UEXITS. If your current exit table already has FastLIST installed, you only need to add an entry for exit 7 as below and reassemble.

If your current Endevor implementation does not include any user EXITS, build a CIUEXITS input table as per your Endevor documentation. Modify the JCL below to include your input and submit. The C1UEXITS table must reside in the Endevor authorized library.

The table lists all the program names to enter into the exit table. The list entry, CIGFOPT7, is an optional exit for notifying approvers of the package in the cast of a WARN, RESOLVE, or FAIL element collision event. The source to this exit is provided in the SAMPLIB data set for further modification, should your installation need additional or different approver notification.

There is a sample of a user exit table input, C1UEXITS, in the Endevor 'JCLLIB' data set delivered with the product.

Exit	The Package Utilities Module
2	CIGFEXEC
3	CIGFEXEC
5	CIGFEXEC
6	CIGFEXEC
7	CIGFEXEC
7	CIGFOPT7 ***

```
//**(JOBCARD)
//*-----------------------------------------------------------------------*
//* NAME: CIGJCL06 - BUILT THE C1UEXITS MODULE                            *
//*                                                                       *
//* THE PURPOSE OF THIS JCL IS TO BUILD A C1UEXITS TABLE WHICH            *
//* INCLUDES THE PACKAGE UTILITY  COLLECTOR MODULES. REFER TO
//* THE ENDEVOR EXITS MANUAL FOR
//* INFORMATION ON HOW TO BUILD THE C1UEXITS INPUT.                       *
//*                                                                       *
//* MODIFY JCL 1) ADD A JOB CARD                                          *
//*            2) MODIFY THE SYSIN STATEMENT TO POINT TO THE DATASET  *
//*               CONTAINING YOUR C1UEXITS INPUT. NOTE THAT INSTREAM  *
//*               INPUT MAY ALSO BE USED INSTEAD OF A DATASET.        *
//*            3) THE SYSLIB DATASET SHOULD POINT TO THE ENDEVOR      *
//*               PROVIDED SOURCE LIBRARY.                            *
//*            4) THE SYSLMOD DATASET SHOULD POINT TO YOUR ENDEVOR    *
//*               AUTHORIZED LIBRARY CONTAINING YOUR CIGINI MODULE.   *
//*-----------------------------------------------------------------------*
//ASM      EXEC PGM=IEV90,
//             REGION=3072K,
//             PARM='NODECK,OBJECT,NOTERM,LIST,XREF(SHORT)'
//SYSLIB   DD  DISP=SHR,DSN=QUAL1.QUAL2.SOURCE
//SYSLIN   DD  DSN=&&SYSLIN,
//             UNIT=WORK,
//             SPACE=(TRK,(3,5)),
//             DISP=(NEW,PASS,DELETE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPUNCH DD  DUMMY
//SYSUT1   DD  UNIT=WORK,SPACE=(TRK,(5,15))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DSN=YOUR.INPUT(C1UEXITS),DISP=SHR
//*-----------------------------------------------------------------------*
//* STEP 2: LINK EDIT THE C1UEXITS TABLE. RE-ASSEMBLE FOR EACH RELEASE*
//*-----------------------------------------------------------------------*
//LINK     EXEC PGM=IEWL,
//             REGION=2048K,
//             PARM='LIST,NCAL,XREF,LET,RENT,REUS',
//             COND=(0,NE)
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSN=&&SYSLIN,
//             DISP=(OLD,DELETE,DELETE)
//SYSLMOD  DD  DSN=QUAL1.QUAL2.AUTHLIB(C1UEXITS),DISP=SHR
//SYSUT1   DD  UNIT=WORK,SPACE=(TRK,(5,15))
```

Figure 3.8
CIGJCL06

Step 8: Optionally, Offload Old Audit Information

Offloading Historical Data

You have an option of offloading all or select executed and/or committed packages from the Endevor package file to the Package Registry. The reason for offloading would be to decrease the size of the actual package file but maintain audit data. To offload previously executed packages you need only to run an Endevor Package ARCHIVE action against existing packages. The Package Utilities will intercept the process and build an audit trial for the old package. For more information on how to run the Endevor Package ARCHIVE, see the Endevor Batch Packages manual. Exits must be enabled to perform this offload task.

Step 9: Optionally, Install a REMAKE Exit Program

The USER PROGRAM Facility

The REMAKE option provides the facility for reusing existing, executed packages. One of the CIGINI options is “USER PROGRAM BEFORE REMAKE = ‘***pgmname***’.” This is a user exit point that allows for customization and control of the REMAKE process. The base product comes with an assembled program called ‘TESTPGM. In its delivered form, this program is only a verification mechanism to make sure that a REMAKE exit is being invoked and should not be used without further customizing.

To build your own REMAKE exit modify the source code included in the SAMPLIB from the tape, member ‘TESTPGM. The things that you can do at the REMAKE exit point include:

1. Change the name of the package to be rebuilt. If you provide a new name, the new named package will be defined and cast in Endevor, based on attributes and SCL of the old named package. If you wish to both change the name and commit/delete the old package data, then you can also specify this in the $USRDS block. Note the commit and delete SCL for the old package ID name will be built but not executed by the standard JCL. Reference *The User Program Facility* for more information on special handling of renamed package ids.

2. Change any standard “DEFINE” attributes of the package. The block sent to you will contain all basic attributes of the package. Package description, sharable options, appends options, and backout options. You can modify these simply by changing the values in the block and returning.

3. Provide an override IMPORT DD for SCL. The default IMPORT DD for SCL is CIGSCL01. This is the ddname that the utility uses to rebuild the SCL from the previous package. If you change the value in the USRSCLDD field, then you are responsible for allocating the DDNAME and providing a new set of SCL for the DEFINE. The new IMPORT DDNAME will be built into the DEFINE STATEMENT.

4. Provide an execution window (date and time). The default on the generated define step to let Endevor set the execution windows.

5. Cancel the REMAKE. If you determine that the package should not be rebuilt, then you should send a non-zero return code back to the remake program. You can control the severity of the return code as well as send back an informational message.

For a complete file layout of the $USRDS and a sample user program see Appendices A and B.

Additional Considerations: Multiple Databases and CIGINI Modules

The ability to support more than one Package Utilities Registry File is supported via the Alternate CIGINI option. A primary CIGINI is always required, but you can have separate Package Utilities Registry Files based on some user-distinction (Userid, application, etc.). This is a customer specific decision and you must have some mechanism for allocating alternate files (a CLIST or separate LOGON PROCS for instance). If you are concurrently installing FastLIST and the Package Utility, you must review *Implementation,* Multiple Databases and CIGINI section in the *FastLIST Administration Guide*.

If you determine that you want more than one Package Utilities Registry configuration, you will need to create a separate initialization module for each additional configuration. Multiple initialization modules can be implemented using the following rules:

1. There is only one active initialization module per session. Like a C1DEFLTS, the initialization module is loaded and used for the life of the application. This includes an Endevor session. You must have some mechanism for allocating different overrides.

2. There must always be a primary CIGINI module in a STEPLIB or job pack area. This module will be used for the PASSWORD check and all of the default initialization parameters.

3. If you want to allow some users to access a different Package Utilities Registry File you must create an override initialization module containing the desired parameters. Remember, packages in the separate databases must be completely distinct. If there is any inventory overlap between databases, the Package Utilities will be unable to keep its information synchronized with Endevor.

4. The alternate initialization modules should be appropriately named. There are no naming standards enforced by the Package Utilities, but the member name must be specifically referenced in the //CIGINI dd allocation.

5. If no //CIGINI allocation is found then the original CIGINI module will be used.

6. If the initialization fails on an override module, the application will be terminated.

The password is checked prior to the attempts to find an override initialization module. No additional password checking is performed even if an override module is found.

Creating the Alternate CIGINI Override Module

To set up an initialization override module, first modify the syntax in *flhq1.flhq2*.SAMPLIB(CIGINI). Follow the syntax guidelines in Figure 3.3. Once you have set up the syntax file for your CIGINI override module, modify and submit CIGJCL04. You have already used this JCL during install. Modify the SYSLMOD dd to send the initialization override to any load library. You should not copy the override CIGINI file to a steplib or jobpack area. This module is only accessed by including a //CIGINI dd in your Package Utilities and Endevor JCL and CLISTS which points to the load library containing the alternate initialization module. The following is an example of how to include an alternate CIGINI module in your JCL.

CIGINI example: //CIGINI   DD DSN=YOUR.LOADLIB(OVERRIDE)

```
//**(JOBCARD)
//**
//*--------------------------------------------------------------------*
//* NAME:PARSE, ASSEMBLE AND LINK CIGINI OR OVERRIDE MODULES           *
//*                                                                    *
//* MODIFY THE JCL IN THE FOLLOWING WAYS.                              *
//*            1) ADD A JOB CARD                                       *
//*            2) THE SYSLMOD DATASET MUST POINT TO YOUR LOADLIB OR    *
//*               CIGINI OVERRIDE DATASET.  YOU MUST COPY YOUR PRIMARY*
//*               CIGINI FILE INTO YOUR STEPLIB OR LINKLIST.  AN       *
//*               OVERRIDE CIGINI FILE IS OPTIONAL.                    *
//*            3) CHANGE flqh1, flqh2 AND dunit AS PER YOUR            *
//*               INSTALLATION WORKSHEET.                              *
//*--------------------------------------------------------------------*
//*--------------------------------------------------------------------*
//* STEP 1: PARSE CIGINI SYNTAX.  BUILD INPUT FOR ASSEMBLER.           *
//*                                                                    *
//*--------------------------------------------------------------------*
//PARSE    EXEC PGM=ICOMPILE
//STEPLIB  DD  DSN=flqh1.flqh2.LOADLIB,DISP=SHR
//CIGIN    DD  DSN=flqh1.flqh2.SAMPLIB(CIGINI),DISP=SHR
//CIGOUT   DD  DSN=&&TEMP,DISP=(NEW,PASS),
//            SPACE=(10,10),UNIT=dunit,
//            DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGLOG   DD  SYSOUT=*
//*--------------------------------------------------------------------*
//* STEP 2: ASSEMBLE THE CIGINI INPUT CREATED IN STEP 1.               *
//*                                                                    *
//*  NOTE: CHOOSE THE DESTINATION OF YOUR CIGINI FILE.                 *
//*                                                                    *
//*--------------------------------------------------------------------*
//ASM      EXEC PGM=IEV90,
//             REGION=3072K,
//             COND=(0,NE),
//             PARM='NODECK,OBJECT,NOTERM,LIST,XREF(SHORT)'
//SYSIN    DD  DSN=&&TEMP,DISP=(OLD,DELETE)
//SYSLIN   DD  DSN=&&SYSLIN,
//             UNIT=dunit,
//             SPACE=(TRK,(3,5)),
//             DISP=(NEW,PASS,DELETE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPUNCH DD  DUMMY
//SYSUT1   DD  UNIT=dunit,SPACE=(TRK,(5,15))
//SYSPRINT DD  SYSOUT=*
//*-------------------------------------------------------------------*
//* STEP 3: LINK EDIT THE CIGINI MODULE                               *
//*                                                                   *
//*  NOTE: CHOOSE THE DESTINATION OF YOUR CIGINI FILE. IF YOU ARE     *
//*        PLANNING ON USING AN INITIALIZATION OVERRIDE MODULE,       *
//*        FIRST BUILD A CIGINI THAT RESIDES IN A STEPLIB DATASET.    *
//*-------------------------------------------------------------------*
//LINK     EXEC PGM=IEWL,
//             REGION=2048K,
//             PARM='LIST,NCAL,XREF,LET,RENT,REUS',
//             COND=(0,NE)
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSN=&&SYSLIN,
//             DISP=(OLD,DELETE,DELETE)
//SYSLMOD  DD  DISP=SHR,DSN=qual1.qual2.LOADLIB(CIGINI)
//*YSLMOD  DD  DISP=SHR,DSN=flhq1.flhq2.CIGINI(OVERRIDE)
//SYSUT1   DD  UNIT=dunit,SPACE=(TRK,(5,15))
```

Figure 3.9
CIGJCL04

Chapter 3 Package Exits and Collision Management

This chapter contains:

- A high-level discussion of how the Package Utilities Exits will affect Endevor Package and Action processing.
- A further explanation of managing collisions through exit processing.
- Importance and difference between Endevor Status and Utility Status Settings.
- A walk through the package life cycle - interfacing with the Package Utilities.

Understanding the Exits

Overview

The Package Utilities is a set of exits and utilities designed to work inside and around Endevor. The exits act as control and collection agents, driving decisions based on user input and Endevor status data.

With each invocation of a Package Action, data is either analyzed or stored for use by the Package Utilities. The Package Utilities exits are built using the standard published package and action user exit facility. These external interfaces, combined with the Chicago Interface Group program infrastructure, provide a robust and highly efficient communication, control, and collection mechanism.

The following pages detail the major events that occur during Package Utilities Exit Processing. If you also have FastLIST installed or are installing FastLIST along with this product, note that all Exit 5 and Exit 6 work is performed once on behalf of all products using the infrastructure.

Following the exit description section will be a series of matrixes that correlate data in the Package Utilities Registry File, Endevor package status information, and the rules of the Package Utilities Registry File.

Remember that this product was designed as a drop-in. This means that there is no loading of historical package information. You start to use the product at what ever stage the package ID is in currently. Thus, packages will be hooked into the system at different stages of the development life cycle. The rule is simple, regardless of the package action, if the package ID is not registered at the start of the action (cast for instance), it will be registered to the Utility and the action will be completed. If the package ID is already registered (there is a header record) then the utility status values are used to determine the validity of the package action.

A High-level Overview of Exits 1-6

Exit 5 - Initialization

The Endevor exit facility loads and executes CIGFEXEC, as per the C1UEXITS table. This program performs the following functions once per Endevor invocation:

- Loads the CIGINI module and checks for an override CIGINI module, loading if found.
- Loads all the exit programs.
- Performs password check.
- Loads the application from the loadlib in the CIGINI module.
- Allocates and opens the database defined in the CIGINI or the override module.
- Allocates and anchors storage for duration of Endevor session.
- Establishes Package Utility execution framework.

If an Endevor exit 5 fails, Endevor initialization fails. The following conditions will cause initialization to fail on behalf of the Package Utility:

- Invalid password
- Database allocation error
- Application loadlib allocation error
- Incorrect WORK or VIO value

Note that an installation may have more than one exit 5 program in its exit tables. If this is the case, then some other exit 5 program may cancel Endevor initialization.

If FastLIST or the Package Utilities initialization fails, the system prints error messages in the CIGOUT data set. Make sure you have allocated a CIGOUT ddname to your foreground and batch Endevor sessions. In foreground, any error messages are written to the terminal screen.

Exit 6 - Delete and Cleanup

The following cleanup tasks are executed at the termination of each Endevor session:

- De-allocate any remaining storage
- Close and de-allocate any files

Exit 2 - The Before Action Exit

The purpose of exit 2 is to check for element collisions with elements being acted upon outside of a package. If the action (except for the **MOVE** action) is being executed from a package, then exit 2 processing is skipped. However, if the element is a stand alone action, then the element is checked to see if it is registered to a package. If it is registered, then the utility fails the action or warns the user based on CIGINI options. Note that currently all actions are affected except **LIST** and **PRINT**. Also note that the **RETRIEVE** action is always treated as a warning, regardless of collision settings.

The following conditions will cause the Package Utility exit 2 to fail:

- An element collision with a Fail or RESOLVE CIGINI option
- If target of **MOVE** action is registered (whether the **MOVE** is in a package or not in a package)

Exit 3 - The After Action Exit

Exit 3 builds and collects vital data to be passed to the "after" execution processing. The log for the package is also updated from exit 3.

A High-level Review of Utility Package Exits

The Endevor package interface is enabled for many exit points, many of which are strictly for logging. The following matrix details each exit point and what is performed.

PACKAGE EXIT	WHEN	FUNCTION
CREATE	Before/After	REMAKE control, status and log updating.
MODIFY	Before/After	REMAKE control, status and log updating.
MID-CAST		Element Collision Management. If package failed here, the CAST fails. If OK or RESOLVE, register elements.
After CAST		Logging.
Review (approve/deny)	Before	RESOLVE checking.
Review (approve/deny)	After	Logging.
Reset	After/Before	Logging. De-register elements.
Exec	Before	RESOLVE and REMAKE checking.
Exec	After	Logging. De-register elements. AUTO-REMAKE launch if option set.
Backin	After	Logging.
Backout	After	Logging.
Commit	After	Logging.
Delete	After	Logging.
Archive	Before	Logging and loading, if not yet loaded.
Ship	Before	Logging.

The following conditions, at specific exit points, will cause the Package Utility exit 7 to issue a cancel action request:

- An element collision with a Fail CIGINI option at Mid-Cast.
- A Create request against an existing registered package, not in REMAKE status.
- Any before action will fail if an IN-DELETION is encountered.
- A Review request will fail if the Utility Status is set to Resolve.
- An Execute request will fail if the Utility Status is set to Resolve.

An informational message will accompany any cancel action request.

Endevor Status and Utility Status Values

Endevor package processing controls what actions can be performed against a certain package by the value of the status field. These values should be familiar to the end user - IN-EDIT, APPROVED, etc. If you request to perform an action and the package is not in the proper status, then Endevor will not allow the action. Likewise, the Package Registry has a Utility Status that is checked at various points. It is often combined with the regular package status for decision making. The following is list of valid values for the Utility Status field:

STATUS	MEANING
BLANK	No Package Utility process in progress. This setting is required for many exit points.
IN-EXECUTION	When a package is being executed, we set the header status to IN-EXECUTION, so that the package ID can not be used for a REMAKE or DELETE UTILITY action.
REMAKE	A REMAKE is in progress. This value will get cleared after the package has been recycled through the DEFINE step.
REMAKE-FAIL	A REMAKE has failed. Can be re-executed.
IN-DELETION	The Package Utility is deleting the package ID from the Registry File. No other actions can be performed against this package ID.
RESOLVE	The mid cast action has intercepted an element collision between this package and another package(s). All have been set to the RESOLVE status. To be able to Approve or Execute this package, the package(s) must be reset and collisions resolved. Or optionally, you can execute the Package Utility RESETID action and clear the status field. Note this action will be logged.
WARNING	The mid-cast action has intercepted an element collision between this package and another package(s). All have been set to the WARNING. This status will not effect the further execution or approval of the package. Note the status will be cleared after execution.

Figure 4.1
Possible Utility Status Settings

Managing Element Collisions via Exits

Detecting Element Collisions

One of the major features of this product is the management of element collisions as per user options. As discussed in earlier chapters, *you* decide what to do in the case of an element collision. This is coded into the CIGINI module and accessed during the MID-CAST exit point. Note that managing collisions is more than absolutely preventing them. In some cases, the collision is acceptable and will not cause outage. The Utility has been designed to offer you a fair amount of flexibility in this area. Between the CIGINI options and the RESETID command, you can customize the process to meet your needs.

Resolving Element Collisions

Once an element collision has been detected by the Package Utility, you have a few options. If you have chosen the “Fail” option on element collisions, then the only package affected will be impacted. You should analyze the elements in the package using the REPORT and REPORTX functions. Look at the package that caused the collision and decide which elements go where. RECAST the package with the collisions resolved. The package CAST will fail until there are no more collisions.

If you have chosen the “Resolve” option on element collisions, then you have two options: 1) Perform an Endevor RESET for all packages set to ‘RESOLVE’, fix the collisions, and re-CAST all packages, or 2) Perform an Endevor RESET for only those packages that need to be modified and then Re-CAST. For the packages that will not be modified, you can execute a Package Utility RESETID command that will clear the Resolve Field. See chapter 7, for RESETID syntax and using the CIGPKUT1 utility program.

Avoiding Production Overlays

Another aspect of collision management is detecting and preventing element overlays during move and transfer processing. All TRANSFER action targets that are Endevor locations will be registered at CAST time. Thus not only is the source monitored, but also the target. Since the destination of the MOVE action is not determined until execution time, the target of a MOVE action will be checked for collisions during exit 2. If the target of the MOVE action is registered to another package, the move action will be canceled or issue a warning, as per the CIGINI settings for collisions. Note: the cancel of the move action within a package will fail the package. This is true for MOVE actions inside or outside of a package.

Understanding Element Collisions

Before making the decision about what to do in the case of an element collision, it is imperative that you know what an element collision is. An element collision condition occurs when an element from a particular stage is **CAST** into a package, only to be **CAST** into subsequent packages or accessed outside of a package via action processing. Consider the effects of the following scenario:

User #1 retrieves, changes, and adds back an element into Endevor. This change is part of a set of changes that must be installed together. The package is scheduled for Friday night. Today is Wednesday.

User #2 regenerates the element in place in response to a change in a macro, then includes the module in a package to contain all programs affected by the macro changes. This package gets executed Thursday night.

The net effect is two-fold: one, an element got moved up the map without the rest of the required changes and two, the Friday night package will fail because an element is missing.

Managing element collisions is key to managing and living with varied development schedules and personnel. The following table outlines what happens if an element collision occurs, per option:

COLLISION OPTION	Non-package Actions and Move Targets	Non-package Retrieves	Package CAST
IGNORE	Do not look for collisions.	Do not look for collisions.	Do not look for collisions.
WARN	Continue with warning, but issue message to action job and cut log record for package with collision.	Warn the user that the element is registered to another package.	Warn the creator and the last update ID that a collision has occurred. Also, create a log record for each package involved in the collision and set the Utility Status to WARN.
RESOLVE	Cancel action and cut log record for package with collision.	Warn the user that the element is registered to another package.	Warn the creator and the last update ID that a collision has occurred. Also, create a log record for each package involved in the collision and set the Utility Status to RESOLVE.
FAIL	Cancel action and cut log record for package with collision.	Warn the user that the element is registered to another package.	Notify the creator and the last update ID that a collision has occurred. Create a log record for the package involved in the collision and set fail the package being **CAST**.

You have to decide which option is best for your installation.

General Notification and Collision Messages

If a WARNING, RESOLVE OR FAIL situation occurs, you will be notified of the collision in three ways. For the active job, a message will be sent to the C1MSGS1 log, the creator userid and last update userid will notified via a MVS SEND command, and the logs of all affected packages will be updated with detailed information about the collision.

The following is an example of C1MSGS1 output and MVS SEND notification issued on a collision (note that CAST is successful but the utility forces a high return code to signal the user of the collision):

```
PKMR400I  BEGINNING ACTION VALIDATION AND SEARCH FOR APPLICABLE APPROVER GROUPS
PKMR401I  ACTION VALIDATION COMPLETED WITHOUT ERRORS
PKMR402I  NO APPROVER GROUP(S) FOUND APPLICABLE FOR PACKAGE
C1U0000I   EXIT 7 - 11:59:38   PKG3151I   PACKAGE SET TO 'RESOLVE' STATUS DUE TO
ELEMENT COLLISIONS .
PKEX216W  WARNING: INVALID CANCEL REQUEST SENT FROM EXIT FUNCTION CAST/AFTER.
ENMP092I  Error message(s) issued during cast of 'WEEKLY0300'
ENBP023I  The CAST action has completed for package ID 'WEEKLY0300'.  Return Cod

ENBP010I  Processing is complete.  Highest return code is 12
```

Figure 4.2
C1MSGS1 Output

The creator of the package and the userid that last updated the package will be notified of collision.

```
RESOLVE SET - COLLISION BETWEEN PKGS DAILY091095 AND WEEKLY0300.
```

Figure 4.3
Collision Message

Additional Approver Notification

In addition to the LOG records per package/per collision and a message sent to the current job log, you may also want to notify the approvers of the package being sent. To perform user notification, include the CIGFOPT7 module as an exit 7 entry in the Endevor exit table. If a collision is detected then all approvers will be notified. This program is delivered as a regular part of the product, but the source is also provided, should a site have different notification requirements. Note: the CIGFOPT7 entry in CIUEXITS should follow the CIGFEXEC entry.

Return Codes From Collision Processing

RC	Meaning
00	The package CAST was successful and no collisions were detected.
08	A collision was detected and the package was set to RESOLVE or WARNING, based on settings.
12	**CAST** was canceled due to error or element collision. Check log for other messages.

A Walk through the Package Life Cycle

In this section, the typical package life cycle will be examined as a function of the Package Utility. It will be presented as a sequence of events.

1. You **Create** Package PROD0695. The package ID will be registered with to the Package Utilities Registry File with a LOG record written for the package ID.

2. You **Modify** Package PROD0695. A LOG record is cut for the package ID.

3. You **Cast** Package PROD0695. At mid-cast, each element is checked for collisions as per user options. The package is either completed or ends with errors. If collision options set at WARN or RESOLVE, all elements are registered to the package. If collision set to FAIL, nothing is registered.

4. You **Approve** Package PROD0695. If Utility Status is blank (not resolve) then a LOG is cut for the package ID. If Utility Status is RESOLVE, then approval action canceled.

5. User **Executes** Package PROD0695. If the Utility Status and Endevor Status are correct, then the package gets executed, elements are de-registered, and LOG records are cut. If AUTO-REMAKE is active, then a REMAKE occurs. (see chapter 5) At the end of the REMAKE, the LOG will reflect that the PROD0695 was **COMMITTED, DELETED, REBUILT, CAST.**

Data Collected and Reporting

Package Utilities Reports chapter details the reports that are available to be run against the log and element data being collected. At any given time, you should be able to report, for instance, a list of all packages that are in RESOLVE STATUS. Refer to *Package Utilities Reports* chapter for report formats and syntax.

Chapter 4 Reusable Packages: CIGPKUT2 - REMAKE and DELETE Command

This chapter contains:

- A detailed walk through of the REMAKE process
- Description of the REMAKE syntax and CIGPKUT2 utility
- An explanation of the JCL for executing a REMAKE
- Setup issues for AUTO-REMAKE processing
- Error conditions and return codes
- Executing a User Program to change the attributes of the new package
- Description of the DELETE syntax and CIGPKUT2 JCL

The REMAKE Utility

Overview

The REMAKE utility is a facility that allows you to reuse your executed package ids, assuming that the package was executed while the Package Utility is active. The net result of the REMAKE request is that the package ID is COMMITTED, DELETED, DEFINED, and CAST automatically, carrying forward the SCL from the previous package and the attributes of the previous package. The package is essentially recycled through the Endevor package process ending up in the CAST phase. Although the package ID has been recycled in Endevor, historical log information exists in the Package Utilities log so that previous activity is not lost.

Promotion Package Exception

Endevor 12.0 and higher users only: Promotion Packages are not eligible for remake processing. If a Promotion Package is processed and auto-remake is enabled, the exits will issue a warning message that the remake has been denied and processing will continue.   If a user requests a remake against a Promotion Package using the stand along utility CIGPKUT2, the utility will issue an error message, an RC=12, and processing will stop.

REMAKE Utility Questions and Answers

- **Where does the REMAKE Utility get the SCL for the new package?**

Throughout the Package Utility monitoring, key pieces of data are captured and stored for later use. For example, per action, the element and option data is stored. When a REMAKE request is processed, the system reviews the actions and elements and rebuilds the SCL as per the action option settings in the CIGINI file or as per override options set at execute time. For instance, you may set MOVE = MOVENEXT as an option. This means that at REMAKE time, take the previous target and turn it into the source of the MOVE action.

The use of these options controls the reuse of certain actions. For instance, perhaps a package had some DELETE actions included. Would it make sense to DELETE the element again? Maybe, but probably not. In that case, the CIGINI option would read DELETE=DISCARD, which means do not recycle DELETE actions.

- **Is it possible to provide other SCL to the package?**

You can completely replace the SCL contained in a package by providing an alternate ddname and allocating that alternate ddname in the REMAKE JCL stream. In the CIGINI file there is an option for a user program to be called. The user program will be passed a block of data, showing the current attributes of the package (ID, comment, sharable, backout?, append, etc.). Upon return to the REMAKE program, if the ddname is not CIGSCL01, then the Package Utilities will assume that you have coded and allocated an alternate ddname containing the new SCL.

- **Where does the REMAKE process get the package attributes to build the package?**

  The Package Utilities maintains a header record for each package ID registered to the utility. In that header record are the current attributes of the package ID.

- **Is it possible to override or change the attributes of a package?**

  As in the case of SCL, you can override any of the DEFINE verb attributes via the user program invoked during the REMAKE exit process. The REMAKE process will use the attributes returned in the $usrds block from the exit. It is in this exit that the user can also cancel the REMAKE.

- **How does the package ID get recycled through Endevor packages?**

  An Endevor package has a predefined life cycle that must be performed in sequence. During the REMAKE process Endevor Package SCL will be written to COMMIT, DELETE, DEFINE, and CAST the package. This is what is meant by ‘recycling’ the package ID. This SCL is then passed to the second step in the job for execution.

- **What happens to the previous package execution audit trail?**

  The package utility maintains a log for each package ID until the package ID is deleted from the Package Registry File. For each package and element action executed against the package, there is an log entry created. This log data can be reviewed, printed and archived on request.

- **How is a REMAKE invoked?**

  There are two ways to invoke a REMAKE. Both utilize exactly the same JCL and syntax, however, one way is to automatically build and submit a REMAKE after a successful package execution, and the other is to manually submit the request.

  To install automatic REMAKE option, you must code the AUTOMATIC-REMAKE option in the CIGINI file. The default in the CIGINI is NO AUTOMATIC-REMAKE. There is also some modification to the Endevor Package JCL to install internal reader and a JCL shell. Check site standards or internal reader submissions.

- **How many times can a package be reused?**

  A package ID can be reused indefinitely. It is up to the user to decide when to stop using the ID or when the package is at the top of the map.

- **What is the difference between a remake and the use of the Endevor Promotion Package, new with Endevor 12.0?**

  The Endevor Promotion Package accomplishes some of what a package remake does. However, when using the Package Utilities remake function, a separate audit log exists, even if the Endevor package has been deleted. Thus allowing for better audit and reporting history.

REMAKE Diagrams

Figure 5.1 shows what happens during STEP1 of a REMAKE process.

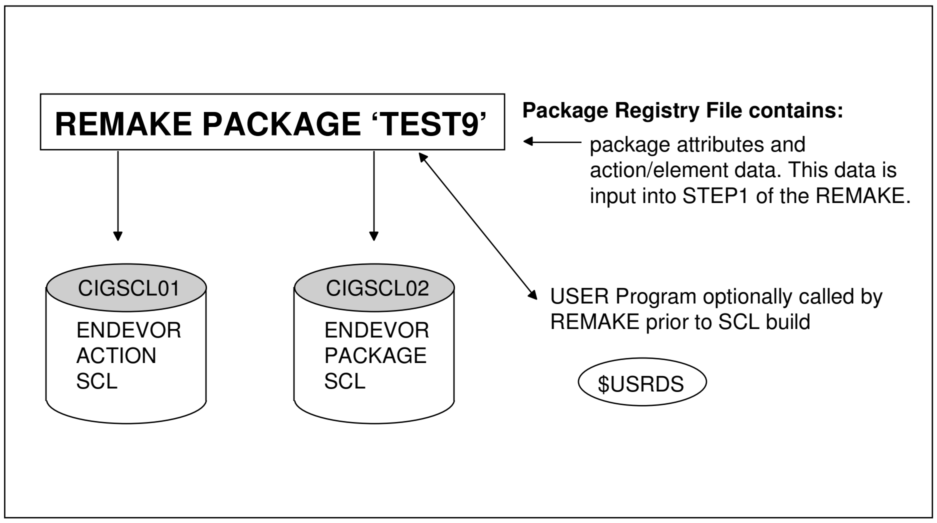

REMAKE PACKAGE 'TEST9'
Package Registry File contains:
package attributes and action/element data. This data is input into STEP1 of the REMAKE.
CIGSCL01
ENDEVOR ACTION SCL
CIGSCL02
ENDEVOR PACKAGE SCL
USER Program optionally called by REMAKE prior to SCL build
$USRDS

Figure 5.1
STEP1 of REMAKE Process

Figure 5.2 shows what happens during STEP2 of a REMAKE process. It is a standard Endevor batch package job.

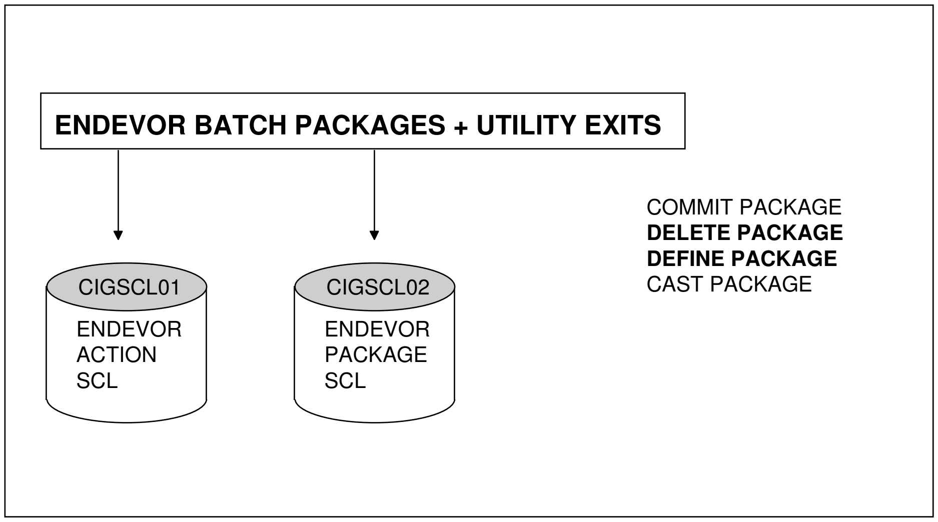
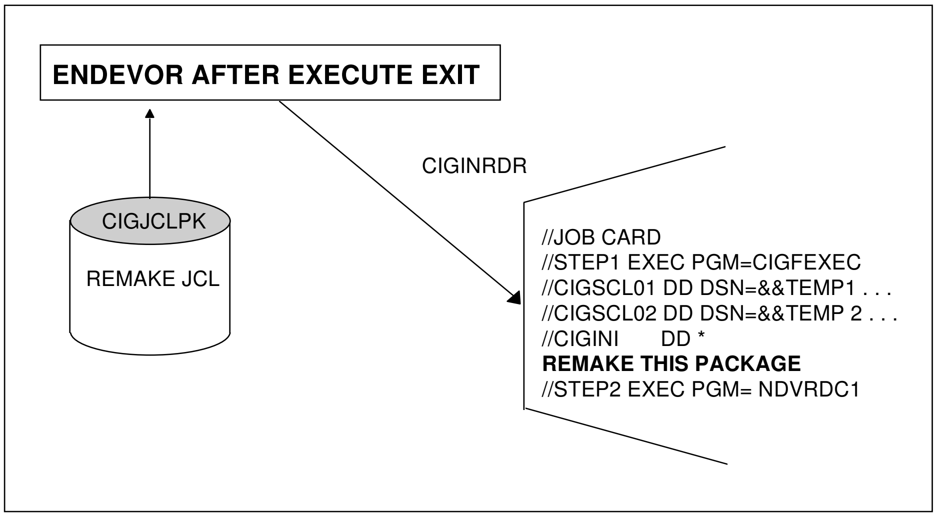


Figure 5.2
STEP2 of REMAKE Process

Figure 5.3 shows what happens at the 'After Execute' Endevor exit if AUTO-REMAKE is installed. The exit merges package information with the shell JCL pointed to by CIGJCLPK and then writes out the job to the INTERNAL reader.



```
//JOB CARD
//STEP1 EXEC PGM=CIGFEXEC
//CIGSCL01 DD DSN=&&TEMP1 . . .
//CIGSCL02 DD DSN=&&TEMP 2 . . .
//CIGINI      DD *
REMAKE THIS PACKAGE
//STEP2 EXEC PGM= NDVRDC1
```

Figure 5.3
Endevor AFTER EXECUTE EXIT

REMAKE JCL

Figure 5.4 contains the JCL used to execute the REMAKE request. This JCL can be found in the JCLLIB loaded from the install tape, member name CIGJCL56. Note that this JCL is also used to perform a Package Utility Delete request. See the last section of this chapter for more information on how to delete packages from the Package Registry File.

The JCL is a two-step process. STEP1 processes the REMAKE command, builds both element SCL and package SCL into CIGSCL01 and CIGSCL02, respectively. These temporary SCL files are then passed to STEP2 which is an Endevor batch package job. Note the inclusion of the CIG data sets in the Endevor step.

```
//**(JOBCARD)
//**
//* -----------------------------------------------------------------*
//* NAME:    CIGJCL56                                                *
//* PURPOSE: EXAMPLE JCL FOR INVOKING PACKAGE REMAKE                 *
//* -----------------------------------------------------------------*
//* -----------------------------------------------------------------*
//* REQUIRED JCL MODIFICATION:                                       *
//*   1) INCLUDE A JOBCARD                                           *
//*   2) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.     *
//*      - FLHQ1 AND FLHQ2                                           *
//*      - QUAL1 AND QUAL2                                           *
//*      - UNIT=TDISK                                                *
//*                                                                  *
//* -----------------------------------------------------------------*
//* FOR PACKAGE WARN MODE UNCOMMENT //C$REMAKE DD DUMMY      Z021213C *
//*                                                                  *
//*   ADD DD STATEMENTS TO ADJUST GENERATED SYNTAX:          Z130709A
//*          //$BOENA   DD  DUMMY     BACKOUT ENABLED
//*          //$CMPVALW DD  DUMMY     VALIDATE COMPONENTS WITH WARNING
//*          //$CMPVALY DD  DUMMY     VALIDATE COMPONENTS
//*          //$CMPVALN DD  DUMMY     DO NOT VALIDATE COMPONENTS
//*          //$SHRABLE DD  DUMMY     SHARABLE PACKAGE       Z140713A
//*   DEFAULT WILL BE:
//*                                   BACKOUT NOT ENABLED
//*                  (7.0 or higher)  INHERIT VALIDATION OF PREV CAST
//*                  (pre 7.0)        SITE DEFAULT BASED ON C1DEFLTS
//* -----------------------------------------------------------------*
//* JCL TO RUN THE REMAKE AND DELETE PACKAGE UTILITY COMMANDS.       *
//* IF NO ENDEVOR SCL WAS WRITTEN THEN THE RETURN CODE WILL BE 4.    *
//* -----------------------------------------------------------------*
//STEP1    EXEC PGM=CIGPKUT2
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGSCL01 DD  DSN=&&TEMP1,DISP=(NEW,PASS),
//             UNIT=TDISK,SPACE=(1,1),
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGSCL02 DD  DSN=&&TEMP2,DISP=(NEW,PASS),
//             UNIT=TDISK,SPACE=(1,1),
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGLOG   DD  SYSOUT=*
//* UNCOMMENT TO ACTIVATE SYNTAX GENERATION OVERRIDE
//*$BOENA   DD  DUMMY     BACKOUT ENABLED
//*$CMPVALW DD  DUMMY     VALIDATE COMPONENTS WITH WARNING
//*$CMPVALY DD  DUMMY     VALIDATE COMPONENTS
//*$CMPVALN DD  DUMMY     DO NOT VALIDATE COMPONENTS
//*$SHRABLE DD  DUMMY     SHARABLE PACKAGE
//CIGIN    DD  *
```

```
** INCLUDE REMAKE COMMAND SYNTAX HERE **
/*
//* --------------------------------------------------------------- *
//* THIS JCL WILL EXECUTE  ENDEVOR PACKAGE SCL BUILT IN STEP ONE.   *
//* FOR PACKAGE WARN MODE UNCOMMENT //C$REMAKE DD DUMMY             *
//* --------------------------------------------------------------- *
//IFSTEP1 IF STEP1.RC = 0 THEN
//STEP2   EXEC PGM=NDVRC1,PARM='ENBP1000'
//STEPLIB  DD  DSN=QUAL1.QUAL2.LOADLIB,DISP=SHR
//CONLIB   DD  DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//C1MSGS1  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//CIGSCL01 DD  DSN=&&TEMP1,DISP=(OLD,DELETE,DELETE)
//ENPSCLIN DD  DSN=&&TEMP2,DISP=(OLD,DELETE,DELETE)
//*C$REMAKE DD  DUMMY
//CIGOUT   DD  SYSOUT=*
// ENDIF
```

Figure 5.4
REMAKE (and DELETE) JCL

PACKAGE SCL Creation Process

The REMAKE program will build the Package SCL so that the package can be reused. The Package SCL creates the following SCL blocks.

COMMIT
DELETE
DEFINE
CAST

The options for the DEFINE and CAST SCL blocks are built from both previous package attributes and site defaults, unless specifically overridden with the use of the override ddnames in the REMAKE step.

The override ddnames are listed below:

DDNAME	Behavior	Default if DD not included
$BOEN	If included, BACKOUT ENABLED will be inserted in the DEFINE and CAST syntax.	Backout will not be enabled
$CMPVALY	If included, VALIDATE COMPONENTS will be inserted in the CAST syntax.	In pre 7.0, if one of the $CMPVAL ddname is not included, then the site default will be used. In 7.0 or higher, the Component Validation setting at the previous CAST will be used.

$CMPVALW	If included, VALIDATE COMPONENTS WITH WARNING will be inserted in the CAST syntax.	In pre 7.0, if one of the $CMPVAL ddname is not included, then the site default will be used. In 7.0 or higher, the Component Validation setting at the previous CAST will be used.
$CMPVALN	If included, DO NOT VALIDATE COMPONENTS will be inserted in the CAST syntax.	In pre 7.0, if one of the $CMPVAL ddname is not included, then the site default will be used. In 7.0 or higher, the Component Validation setting at the previous CAST will be used.
$SHRABLE	If included, SHARABLE PACKAGE will be inserted in the DEFINE syntax.	If not included, then the Endevor defaults will be used.

REMAKE Syntax

Figure 5.5 below contains the REMAKE syntax.

```
REMAKE       PACKAGE        ‘package-name’

OPTIONS
      MOVE           {ASIS | DISCARD | MOVENEXT }
      GENERATE       {ASIS | DISCARD }
      DELETE         {ASIS | DISCARD }
      RETRIEVE       {ASIS | DISCARD }
      TRANSFER       {ASIS | DISCARD }
      ADD            {ASIS | DISCARD }
      UPDATE         {ASIS | DISCARD }
      LIST           {ASIS | DISCARD }
      PRINT          {ASIS | DISCARD }
      ARCHIVE        {ASIS | DISCARD }
      RESTORE        {ASIS | DISCARD }
.
```

Figure 5.5
REMAKE Syntax

Syntax Rules

Only the first phrase is required, where ***package-name*** is equal to a fully qualified, successfully executed, package ID. The name should be enclosed in quotes.

Only one REMAKE statement allowed per execution of CIGPKUT2. If more than one is coded, then the step will fail.

Other options will override the action options settings in the CIGINI file. These are optional.

The syntax block must end with a period.

Syntax Examples

In Example 1, You have requested a REMAKE of package TEST9. All SCL actions will be treated as per the CIGINI settings.

```
REMAKE PACKAGE ‘TEST9’ .
```

Figure 5.6
Syntax Example #1

In Example 2, You have requested a REMAKE of package TEST9. You have also requested that all GENERATE actions be recreated ASIS, overriding whatever is in the CIGINI file.

```
REMAKE PACKAGE ‘TEST9’
OPTIONS  GENERATE ASIS.
```

Figure 5.7
Syntax Example #2

REMAKE Execution Log

Figure 5.8 contains the output from Step1 of the REMAKE job. Note that output will be the same regardless of whether you manually submit the REMAKE job or have the Package Utilities submit the job via the internal reader.

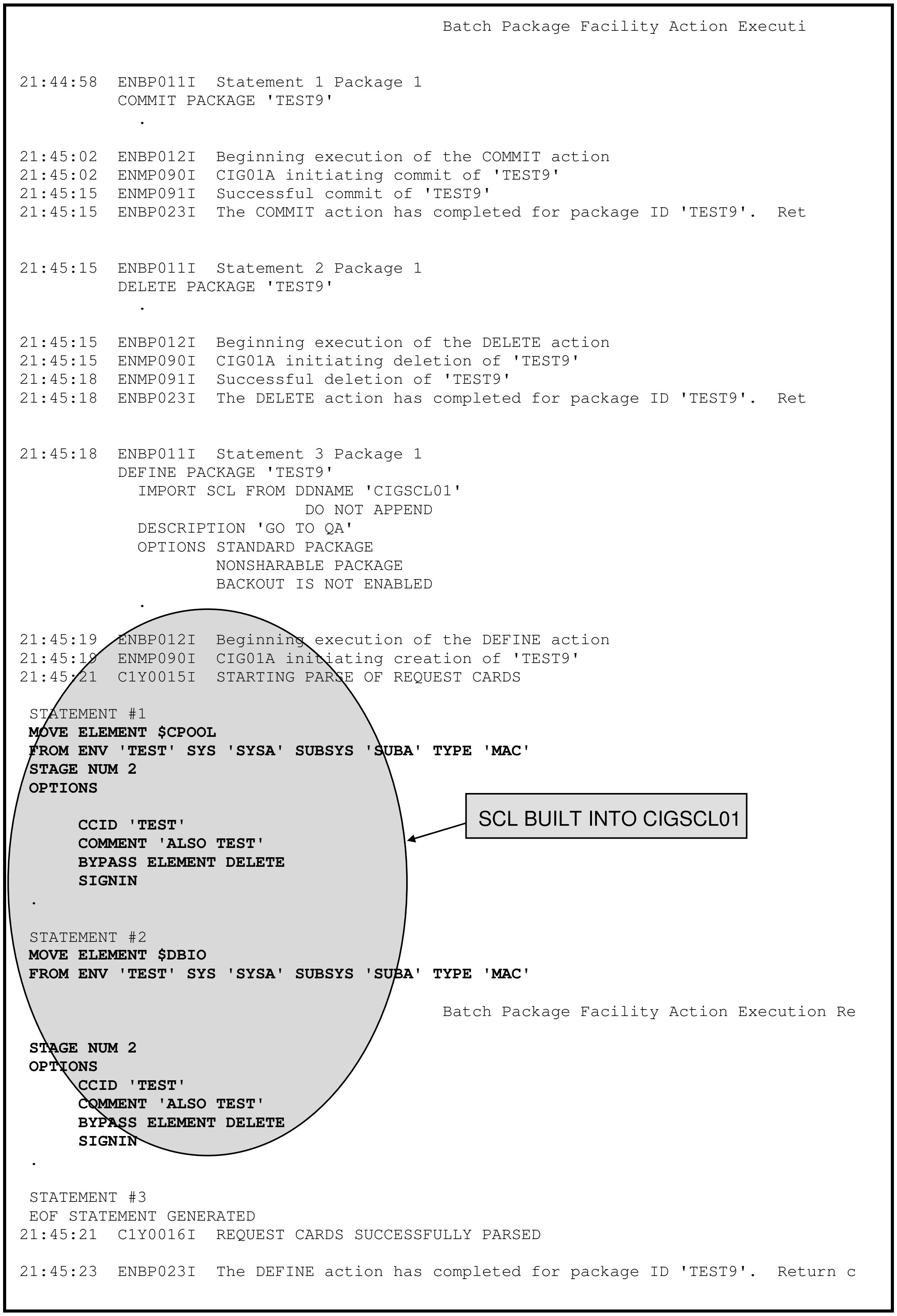
```
21:44:00  FST0281I ----- COMMON INITIALIZATION INFORMATION -----
21:44:00  FST0251I PRODUCT LOAD LIBRARY......... CIGT.XIFR01.LOADLIB
21:44:00  FST0280I ALTERNATE CIGINI ALLOWED?.... N
21:44:00  FST0269I PACKAGE VSAM FILE........... CIGT.PACKAGE.DB

DATE 08/01/13 TIME 21:44:00     P A C K A G E   U T I L I T Y, RELEASE 12.0
                                 E X E C U T I O N  R E P O R T
21:44:01  FST1102I  PARSER BEGINS
21:44:01  FST0020I  REMAKE PACKAGE TEST9  .
21:44:01  FST1103I  PARSER ENDS, RC=0000
21:44:03  PKG3107I  Endevor ELEMENT SCL SUCCESSFULLY REBUILT INTO CIGSCL01 DD, RC = (00).
21:44:03  PKG3109I  Endevor PACKAGE SCL SUCCESSFULLY REBUILT INTO CIGSCL02 DD, RC = (00).
```

Figure 5.8
Messages From Step1 of REMAKE

Figure 5.9 below contains the Endevor output from Step2 of the REMAKE job. Note the package SCL executed was generated in Step1. Note also, that element SCL is being imported from the CIGSCL01 ddname.

```
                                          Batch Package Facility Control Statement S

21:44:57  ENBP900I  Control statement parsing is beginning

21:44:57  ENBP923I  Statement number 1
          COMMIT PACKAGE 'TEST9'
           .

21:44:57  ENBP923I  Statement number 2
          DELETE PACKAGE 'TEST9'
           .

21:44:57  ENBP923I  Statement number 3
          DEFINE PACKAGE 'TEST9'
            IMPORT SCL FROM DDNAME  CIGSCL01
            DO NOT APPEND
            DESCRIPTION 'GO TO QA'
            OPTIONS
            STANDARD PACKAGE
            NONSHARABLE PACKAGE
            BACKOUT IS NOT ENABLED
           .

21:44:57  ENBP923I  Statement number 4
          CAST   PACKAGE 'TEST9'
            OPTIONS
            BACKOUT IS NOT ENABLED
            VALIDATE COMPONENTS
           .
```

SCL BUILT INTO CIGSCL02



```
                                              Batch Package Facility Action Executi


21:44:58  ENBP011I  Statement 1 Package 1
          COMMIT PACKAGE 'TEST9'
            .

21:45:02  ENBP012I  Beginning execution of the COMMIT action
21:45:02  ENMP090I  CIG01A initiating commit of 'TEST9'
21:45:15  ENMP091I  Successful commit of 'TEST9'
21:45:15  ENBP023I  The COMMIT action has completed for package ID 'TEST9'.  Ret


21:45:15  ENBP011I  Statement 2 Package 1
          DELETE PACKAGE 'TEST9'
            .

21:45:15  ENBP012I  Beginning execution of the DELETE action
21:45:15  ENMP090I  CIG01A initiating deletion of 'TEST9'
21:45:18  ENMP091I  Successful deletion of 'TEST9'
21:45:18  ENBP023I  The DELETE action has completed for package ID 'TEST9'.  Ret


21:45:18  ENBP011I  Statement 3 Package 1
          DEFINE PACKAGE 'TEST9'
            IMPORT SCL FROM DDNAME 'CIGSCL01'
                             DO NOT APPEND
            DESCRIPTION 'GO TO QA'
            OPTIONS STANDARD PACKAGE
                    NONSHARABLE PACKAGE
                    BACKOUT IS NOT ENABLED
            .

21:45:19  ENBP012I  Beginning execution of the DEFINE action
21:45:19  ENMP090I  CIG01A initiating creation of 'TEST9'
21:45:21  C1Y0015I  STARTING PARSE OF REQUEST CARDS

 STATEMENT #1
 MOVE ELEMENT $CPOOL
 FROM ENV 'TEST' SYS 'SYSA' SUBSYS 'SUBA' TYPE 'MAC'
 STAGE NUM 2
 OPTIONS

      CCID 'TEST'
      COMMENT 'ALSO TEST'
      BYPASS ELEMENT DELETE
      SIGNIN
 .

 STATEMENT #2
 MOVE ELEMENT $DBIO
 FROM ENV 'TEST' SYS 'SYSA' SUBSYS 'SUBA' TYPE 'MAC'
                                              Batch Package Facility Action Execution Re


 STAGE NUM 2
 OPTIONS
      CCID 'TEST'
      COMMENT 'ALSO TEST'
      BYPASS ELEMENT DELETE
      SIGNIN
 .

 STATEMENT #3
 EOF STATEMENT GENERATED
21:45:21  C1Y0016I  REQUEST CARDS SUCCESSFULLY PARSED

21:45:23  ENBP023I  The DEFINE action has completed for package ID 'TEST9'.  Return c
```

```
21:45:23  ENBP011I  Statement 4 Package 1
          CAST PACKAGE 'TEST9'
            OPTIONS BACKOUT IS NOT ENABLED
                    VALIDATE COMPONENTS
            .
21:45:23  ENBP012I  Beginning execution of the CAST action
21:45:23  ENMP090I  CIG01A initiating cast of 'TEST9'
21:45:24  C1Y0015I  STARTING PARSE OF REQUEST CARDS

 STATEMENT #1
 MOVE ELEMENT $CPOOL
 FROM ENV 'TEST' SYS 'SYSA' SUBSYS 'SUBA' TYPE 'MAC'
 STAGE NUM 2
 OPTIONS
      CCID 'TEST'
      COMMENT 'ALSO TEST'
      BYPASS ELEMENT DELETE
      SIGNIN
 .

 STATEMENT #2
 MOVE ELEMENT $DBIO
 FROM ENV 'TEST' SYS 'SYSA' SUBSYS 'SUBA' TYPE 'MAC'
 STAGE NUM 2
 OPTIONS
      CCID 'TEST'
      COMMENT 'ALSO TEST'
      BYPASS ELEMENT DELETE
SIGNIN
 .

 STATEMENT #3

                                        Batch Package Facility Action Executi

 EOF STATEMENT GENERATED
21:45:24  C1Y0016I  REQUEST CARDS SUCCESSFULLY PARSED

21:45:24  PKMR400I  BEGINNING ACTION VALIDATION AND SEARCH FOR APPLICABLE APPROV
21:45:33  PKMR401I  ACTION VALIDATION COMPLETED WITHOUT ERRORS
21:45:33  PKMR402I  NO APPROVER GROUP(S) FOUND APPLICABLE FOR PACKAGE
21:45:33  PKMR791I  COMPONENT VALIDATION STARTED
21:45:33  PKMR799I  COMPONENT VALIDATION COMPLETED WITHOUT ERRORS
21:45:36  ENMP091I  Successful cast of 'TEST9'
21:45:36  ENBP023I  The CAST action has completed for package ID 'TEST9'.  Return

                                         Batch Package Facility Action Summary


                      Action       Statement  Package  Package           Return
                      Name          Number    Number   Name               Code
                      -----------  ---------  -------  ---------------  -------
                      COMMIT             1         1   TEST9                0
                      DELETE             2         1   TEST9                0
                      DEFINE             3         1   TEST9                0
                      CAST               4         1   TEST9                0
```

Figure 5.9
Endevor Batch Output from REMAKE

Figure 5.10 contains an example of regular and trace messages that will be written to the CIGOUT data set in Step2.

```
13:01:59  FST0703I IN EXIT 5: BEGINNING INITIALIZATION PROCESSING.
13:01:59  FST0281I ----- COMMON INITIALIZATION INFORMATION -----
13:01:59  FST0251I PRODUCT LOAD LIBRARY......... CIGT.XIFR01.LOADLIB
13:01:59  FST0257I WORK UNIT.....................WORK
13:01:59  FST0259I VIO UNIT..................... WORK
13:01:59  FST0284I Endevor CONLIB DSNAME...... SYS2.CIG.ENDEVRC1.CONLIB.B9212C
13:01:59  FST0280I ALTERNATE CIGINI ALLOWED?.... N
13:01:59  FST0282I ---- FASTLIST INITIALIZATION INFORMATION ----
13:01:59  FST0245I FASTLIST PRIMARY VSAM FILE... CIGT.FLSTBIG
13:01:59  FST0246I FASTLIST INDEXED VSAM FILE... CIGT.FLSTBIG.PATH
13:01:59  FST0244I INFORMATION TO COLLECT?...... COMPONENTS(Y)   CCID(Y)
13:01:59  FST0254I FOREGROUND EXECUTION ALLOWED? Y
13:02:03  FST0283I ----- PACKAGE INITIALIZATION INFORMATION ----
13:02:03  FST0269I PACKAGE VSAM FILE............ CIGT.PACKAGE.DB
13:02:03  FST0289I ELEMENT COLLISION SWITCH..... RESOLVE PACKAGES
13:02:03  FST0290I AUTOMATIC REMAKE............. Y
13:02:03  FST0291I LOG RECORDING IN EFFECT?..... Y
13:02:03  FST0292I REMAKE EMERGENCY PACKAGES?... N
13:02:03  FST0293I EXIT PROGRAM NAME............ TESTPGM
13:02:03  FST0294I ACTION OPTIONS
13:02:03  FST0295I      ADD=ASIS             UPDATE=ASIS
13:02:03  FST0295I      MOVE=MOVENEXT       TRANSFER=ASIS
13:02:03  FST0295I      GENERATE=ASIS        DELETE=ASIS
13:02:03  FST0295I      PRINT=ASIS           LIST=ASIS
13:02:03  FST0295I      ARCHIVE=ASIS         RESTORE=ASIS
13:02:03  FST0295I      RETRIEVE=ASIS


13:02:06  FST0702I IN EXIT 5: INITALIZATION PROCESSING WAS SUCCESSFUL.
13:02:17  PKG1202I  IN CIGPKX07 - AT 'AFTER' 'COMMIT' PACKAGE EXIT POINT.
13:02:18  FST0500I  DATABASE QUERY RESULTED IN 0001 'P-HEADER' RECORDS SENT BACK
13:02:22  PKG1202I  IN CIGPKX07 - AT 'AFTER' 'DELETE' PACKAGE EXIT POINT.
13:02:22  FST0500I  DATABASE QUERY RESULTED IN 0001 'P-HEADER' RECORDS SENT BACK
13:02:23  PKG1202I  IN CIGPKX07 - AT 'BEFORE' 'CREATE' PACKAGE EXIT POINT.
13:02:23  FST0500I  DATABASE QUERY RESULTED IN 0001 'P-HEADER' RECORDS SENT BACK
13:02:24  PKG1202I  IN CIGPKX07 - AT 'AFTER' 'CREATE' PACKAGE EXIT POINT.
13:02:24  FST0500I  DATABASE QUERY RESULTED IN 0001 'P-HEADER' RECORDS SENT BACK
13:02:24  FST0500I  DATABASE QUERY RESULTED IN 0001 'ACT-SUM' RECORDS SENT BACK

13:02:24  PKG3173I  REMAKE HAS COMPLETED FOR PACKAGE TEST9 THROUGH DEFINE.
13:02:24  PKG3174I  ALL ACTION SUMMARY RECORDS WILL BE DELETED.
13:02:24  PKG3175I  HEADER RECORD WILL REFLECT CURRENT Endevor STATUS.

13:02:34  PKG1202I  IN CIGPKX07 - AT 'MID' 'CAST' PACKAGE EXIT POINT.
13:02:34  PKG1202I  IN CIGPKX07 - AT 'MID' 'CAST' PACKAGE EXIT POINT.
13:02:34  PKG1202I  IN CIGPKX07 - AT 'MID' 'CAST' PACKAGE EXIT POINT.
13:02:34  FST0500I  DATABASE QUERY RESULTED IN 0000 'ELE-REG' RECO
13:02:34  PKG1202I  IN CIGPKX07 - AT 'AFTER' 'CAST' PACKAGE EXIT POINT.
13:02:34  FST0500I  DATABASE QUERY RESULTED IN 0001 'P-HEADER' RECORDS SENT BACK
13:02:35  FST0012I IN EXIT 6
```

Figure 5.10
Example of Step2 CIGOUT REMAKE Messages

AUTO-EXEC Setup Options

The previous pages of syntax, JCL, and output will be exactly the same for a REMAKE request generated from the 'After Execute' package action. The difference to the user is that they need to modify the Endevor batch package JCL and the Endevor foreground operating environment to include the following ddnames.

The JES2 INTERNAL READER DD. This file will be used to pass the modified JCL to JES2.

//CIGINRDR DD   SYSOUT=(A,INTRDR)

The external user JCL DD. This file should contain the REMAKE shell from the JCLLIB library, member called CIGJCL56. You should modify this JCL to include a JOB card.

//CIGJCLPK DD   DSN=*USER.DATASET*,DISP=SHR

Wherever Endevor packages are invoked, you must make sure that these additional files are allocated. One suggestion would be to add these skeleton JCL cards to the ISPF additional JCL cards currently supported in standard Endevor. Any stand-alone JCL that is maintained and executed outside of Endevor would need to be modified. Also the standard Endevor CLIST for invoking foreground would also need to be enhanced to include these ddnames if you perform foreground package executions.

The User Program Facility

Regardless of automatic or manual submission of the REMAKE request, the REMAKE option provides the facility for reusing existing, executed packages. One of the CIGINI options is "USER PROGRAM BEFORE REMAKE = '***pgmname***'. This is a user exit point that allows for customization and control of the REMAKE process. The things that you can do at this point are:

1. Change the name of the package to be rebuilt. If you provide a new name, the newly named package will be defined and cast in Endevor, based on attributes and SCL of the old named package. If you wish to both change the name and commit/delete the old package data, then you can also specify this in the $USRDS block. Note the commit and delete SCL for the old package ID name will be built but not executed by the standard JCL. See figure 5.8 for an example of JCL changes that would be required to both request and execute the commit and delete SCL for the old package ID.

2. Change or enforce any standard "define" attributes of the package. The block sent to you will contain all basic attributes of the package. Package description, sharable options, append options, and backout options. You can then modify these simply by changing the values in the block and returning.

3. Provide an override IMPORT DD for SCL. The default IMPORT DD for SCL is CIGSCL01. This is the ddname that the utility uses to rebuild the SCL from the previous package. If you change the value in the USRSCLDD field, then you are responsible for allocating the DDNAME and providing a new set of SCL for the DEFINE. The new IMPORT DDNAME will be built into the DEFINE STATEMENT.

4. Provide an execution window (date and time). The default on the generated define step to let Endevor set the execution windows. If you want to set the dates and time, they must be in Endevor format.

5. Cancel the REMAKE. If you determine that the package should not be rebuilt, then you should send a non-zero return code back to the remake program. You can control the severity of the return code and also send back a message to be passed onto the REMAKE log.

Changing the Package ID on a REMAKE - Special Considerations

The system default for a REMAKE is to use the same package ID, committing and deleting the old version, while maintaining the audit trails. You can override this package value with a new name. In this case, the system default is to leave the old package ID in place, and model the new package on the old package ID. You can also request that the old package ID be deleted. In this case, a modification would also need to be implemented with such a request. By setting the USRDCURR field to ‘Y’, the system will build commit and delete SCL into the CIGSCL03 ddname. This ddname will not be processed by step 2 of the REMAKE process. Instead, you will need to either add an additional step to their various JCLs or save the CIGSCL03 file for later processing. Note that CIGSCL03 ddname is not one of the standard ddnames.

Figure 5.11 below shows the additional ddname and step that would be required for skeleton CIGSKL02 and shell CIGJCL56.

```
//* -----------------------------------------------------------------*
//* JCL TO RUN THE REMAKE AND DELETE PACKAGE UTILITY COMMANDS.        *
//* IF NO ENDEVOR SCL WAS WRITTEN THEN THE RETURN CODE WILL BE 4.     *
//* NOTE THE SPECIAL CIGSCL03 DDNAME.
//* -----------------------------------------------------------------*
//STEP1    EXEC PGM=CIGPKUT2
//* CIG PRODUCT LOADLIB
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGSCL01 DD  DSN=&&TEMP1,DISP=(NEW,PASS),
//             SPACE=(1,1),UNIT=WORK,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGSCL02 DD  DSN=&&TEMP2,DISP=(NEW,PASS),
//             SPACE=(1,1),UNIT=WORK,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGSCL03 DD  DSN=&&TEMP3,DISP=(NEW,PASS),
//             SPACE=(1,1),UNIT=WORK,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGLOG   DD  SYSOUT=*
//CIGIN    DD  *
/*
//* ------------------------------------------------------------- *
//* THIS JCL WILL EXECUTE  ENDEVOR PACKAGE SCL BUILT IN STEP ONE.   *
//* ------------------------------------------------------------- *
//IFSTEP1 IF STEP1.RC = 0 THEN
//STEP2   EXEC PGM=NDVRC1,PARM='ENBP1000'
//* ENDEVOR AUTHLIB
//* ENDEVOR CONLIB
//STEPLIB DD DSN=QUAL1.QUAL2.AUTHLIB,DISP=SHR
//CONLIB  DD DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//C1MSGS1  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//CIGSCL01 DD  DSN=&&TEMP1,DISP=(OLD,DELETE,DELETE)
//ENPSCLIN DD  DSN=&&TEMP2,DISP=(OLD,DELETE,DELETE)
//CIGOUT   DD  SYSOUT=*
//*CIGTRACE DD  DUMMY
//      ENDIF
//* ------------------------------------------------------------- *
//* THIS JCL WILL EXECUTE  ENDEVOR COMMIT & DELETE SCL IF THE USER  *
//* HAS REQUESTED BOTH A RENAME AND COMMIT/DELETE OF OLD.           *
//* NOTE USER MUST MODIFY ALL REMAKE JCLS TO INCLUDE THIS STEP      *
//* ------------------------------------------------------------- *
//IFSTEP2 IF STEP2.RC < 12 THEN
//STEP3   EXEC PGM=NDVRC1,PARM='ENBP1000'
//* ENDEVOR AUTHLIB
//* ENDEVOR CONLIB
//STEPLIB DD DSN=QUAL1.QUAL2.AUTHLIB,DISP=SHR
//CONLIB  DD DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//C1MSGS1  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//ENPSCLIN DD  DSN=&&TEMP3,DISP=(OLD,DELETE,DELETE)
//CIGOUT   DD  SYSOUT=*
//      ENDIF
```

Figure 5.11
Remake Exception JCL

For a complete file layout of the $USRDS and a sample user program see Appendices A and B.

REMAKE Return Codes and Meanings

Return Code	Meaning
00	All processes completed successfully
04	If from Step1, this means that no SCL qualified to be rebuilt. If from Step2, check for other Endevor messages.
12	If from Step1, package ID was not available for REMAKE or status settings where not valid. Could also mean a syntax error has occurred. Check log for other messages. If from Step2, check for other Endevor or CIGOUT messages.

All messages sent from the utility will begin with the prefix FST or PKG and will be written out to CIGLOG ddname for Step1 and CIGOUT ddname for Step2.

DELETE JCL

Figure 5.12 contains the JCL that is used to execute the DELETE request. This JCL can be found in the JCLLIB loaded from the install tape, member name CIGJCL56.

The JCL is a two-step process. STEP1 processes the DELETE command, builds package SCL into CIGSCL02. This temporary SCL file is then passed to STEP2 which is an Endevor batch package job. Note the inclusion of the CIG data sets in the Endevor step. Note also that the CIGSCL01 ddname is not required for the DELETE verb.

```
//**(JOBCARD)
//**
//* -----------------------------------------------------------------*
//* NAME:    CIGJCL52                                                *
//* PURPOSE: JCL TO RUN THE REMAKE AND DELETE PACKAGE UTILITY         *
//*          COMMAND VERBS.                                          *
//* -----------------------------------------------------------------*
//*  TO USE THIS JCL, YOU MUST:                                      *
//*         1) INSERT A VALID JOB CARD WITH VALID CLASS              *
//*         2) MAKE SURE THAT THE STEPLIB POINTS TO YOUR Endevor      *
//*            AUTHORIZED DATASET AND INCLUDES THE DATASET THAT       *
//*            CONTAINS CIGINI AND CIGFEXEC.                          *
//*         3) MAKE SURE THAT THE CONLIB POINTS TO YOUR Endevor       *
//*            CONLIB DATASET.                                       *
//*         4) CHANGE FLHQ1, FLHQ2, QUAL1 AND QUAL2 AS PER YOUR       *
//*            INSTALLATION SHEET                                    *
//*         5) CHANGE THE UNIT=WORK TO THE APPROPRIATE VALUE FOR A    *
//*            WORK UNIT.                                            *
//*         6) MAKE SURE THAT IF YOU ARE USING AN OPTIONAL CIGINI     *
//*            DD THAT YOU INCLUDE THE DD STATEMENT IN BOTH STEPS.    *
//* ---------------------------------------------------------------- *
//* JCL TO RUN THE REMAKE AND DELETE PACKAGE UTILITY COMMANDS.        *
//* IF NO Endevor SCL WAS WRITTEN THEN THE RETURN CODE WILL BE 4.     *
//* ---------------------------------------------------------------- *
//STEP1    EXEC PGM=CIGPKUT2
//* CIG PRODUCT LOADLIB
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGSCL01 DD  DSN=&&TEMP1,DISP=(NEW,PASS),
//             SPACE=(1,1),UNIT=WORK,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGSCL02 DD  DSN=&&TEMP2,DISP=(NEW,PASS),
//             SPACE=(1,1),UNIT=WORK,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//CIGLOG   DD  SYSOUT=*
//CIGIN    DD  *
 REMAKE  PACKAGE 'TEST1' .
/*
//* -------------------------------------------------------------- *
//* THIS JCL WILL EXECUTE  Endevor PACKAGE SCL BUILT IN STEP ONE.   *
//* -------------------------------------------------------------- *
//IFSTEP1 IF STEP1.RC = 0 THEN
//STEP2   EXEC PGM=NDVRC1,PARM='ENBP1000'
//* ENDEVOR AUTHLIB
//* ENDEVOR CONLIB
//STEPLIB DD DSN=QUAL1.QUAL2.AUTHLIB,DISP=SHR
//CONLIB  DD DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//C1MSGS1  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//CIGSCL01 DD  DSN=&&TEMP1,DISP=(OLD,DELETE,DELETE)
//ENPSCLIN DD  DSN=&&TEMP2,DISP=(OLD,DELETE,DELETE)
```

```
//* ------------------------------------------------------------- *
//* PACKAGE UTILITY DATASETS                                      *
//* NOTE THE TRACE IS OPTIONAL                                    *
//* ------------------------------------------------------------- *
//CIGOUT   DD  SYSOUT=*
//CIGTRACE DD  DUMMY
```

Figure 5.12
DELETE JCL

DELETE Syntax

Figure 5.13 below contains the DELETE syntax.

```
DELETE

      PACKAGE        ‘package-name’

      WHERE STATUS EQUALS ‘ndvr-pkg-status’

      WHERE UTILITY STATUS EQUALS ‘utility-status’

      WHERE FROM DATE EQUAL ‘YY/MM/DD’ [THROUGH DATE EQUAL
      ‘YY/MM/DD’]

      WHERE OLDER THAN ‘999’ DAYS
.
```

Figure 5.13
DELETE- CIGPKUT2 Syntax

Note that this syntax is exactly the same as the CIGPKUT1 syntax. The minimum required is a verb and a package ID. For example:

```
DELETE PACKAGE ‘TESTXX’  .
```

Where the package is a one to 16 valid package ID value, enclosed in quotes.

‘Where’ Clauses

The ‘where’ clauses allow you to further limit or select packages to be included in the reports. The rules for the where clauses are as follows:

WHERE STATUS EQUALS *status*

This clause will limit the report to packages that meet the stated Endevor status value.

WHERE UTILITY STATUS EQUALS *status*

This clause will limit the verb to packages that meet the stated Package Utility status value. Valid values are:

RESOLVE	packages that have been in a collision
REMAKE	packages that are in the process of a REMAKE
IN-DELETION	packages that are in the process of being deleted from the registry and Endevor .

WHERE FROM DATE EQUALS ‘yy/mm/dd’ [THROUGH DATE EQUALS ‘yy/mm/dd’] .

If only one date parameter is sent, then the search will return packages whose utility last update date is equal to the value. If both dates are returned, then the search will return packages whose utility last update date is within the range of dates, inclusive. The date range must be valid to pass parsing requirements.

WHERE OLDER THAN ‘999’ DAYS.

This clause is mutually exclusive with date ranges. You provide a one- to three-character value from ‘1’ to ‘999’. The search will return packages whose utility last update date is less than today’s date minus the ‘older than’ value.

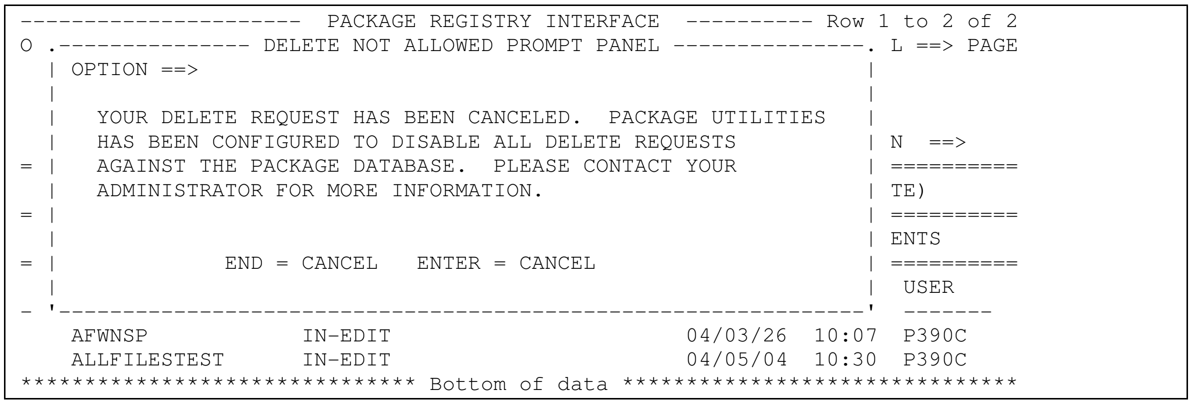
DISABLE DELETE Option

If Package Utilities has been implemented with the DISABLE DELETE option set in the CIGINI, users will not be able to perform a Package Utilities Delete. If they attempt to perform a Delete they will get the following message in the CIGLOG.

```
PKG3304E  DELETE OR CLEARLOG REQUEST DENIED.
PKG3305E  BOTH FUNCTIONS ARE DISABLED FOR YOUR INSTALLATION.
```

Fig 5.14 Disable Delete Messages

If they attempt to request a Delete function from the ISPF front end they will get the following error prompt.



```
----------------------  PACKAGE REGISTRY INTERFACE  ---------- Row 1 to 2 of 2
O .--------------- DELETE NOT ALLOWED PROMPT PANEL ---------------. L ==> PAGE
  | OPTION ==>                                                    |
  |                                                               |
  |   YOUR DELETE REQUEST HAS BEEN CANCELED.  PACKAGE UTILITIES   |
  |   HAS BEEN CONFIGURED TO DISABLE ALL DELETE REQUESTS          | N  ==>
= |   AGAINST THE PACKAGE DATABASE.  PLEASE CONTACT YOUR          | ==========
  |   ADMINISTRATOR FOR MORE INFORMATION.                         | TE)
= |                                                               | ==========
  |                                                               | ENTS
= |             END = CANCEL   ENTER = CANCEL                     | ==========
  |                                                               |  USER
_ '---------------------------------------------------------------'  -------
    AFWNSP            IN-EDIT                     04/03/26  10:07  P390C
    ALLFILESTEST      IN-EDIT                     04/05/04  10:30  P390C
****************************** Bottom of data ******************************
```

Figure 5.15 Disable Delete Prompt

Chapter 5 ISPF Front-end

This chapter contains information on:

- How to use the ISPF front-end.
- How to generate and submit reports.
- How to use line commands to solve collision problems.
- Setting up job card values for file skeletons.

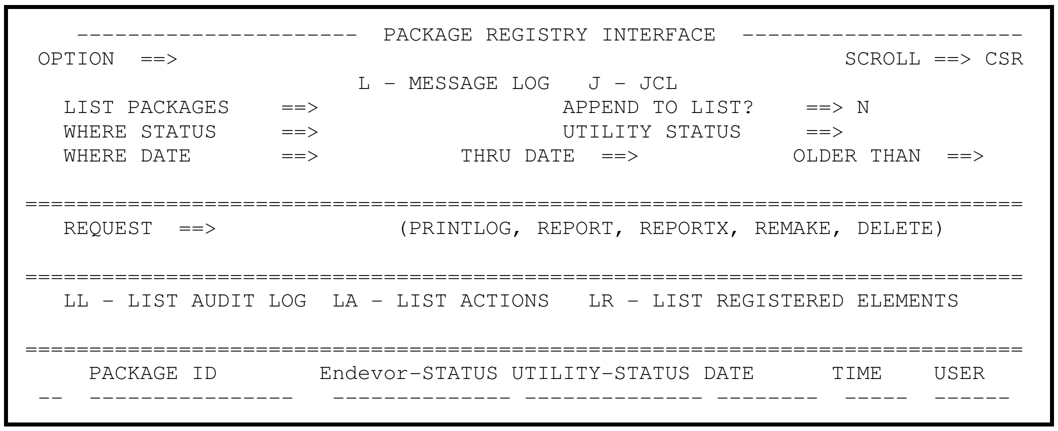
Introduction

Overview

The purpose of the Package Utilities ISPF interface is to allow you to review package information on-line and to generate report JCL and command verbs. There is one main panel to the interface from which you can request a list of packages based on various types of search data.

Figure 6.1 shows the main panel. Note that the top portion of the screen is for entering your search criteria, the bottom portion of the screen is for the package data list, and the middle portion of the screen is for generating report syntax.

From this panel you can list packages based on name or status filtering. You can submit various report request as well as request package REMAKE or DELETE. In addition, the line commands available help you detect and resolve collisions.



```
    ----------------------  PACKAGE REGISTRY INTERFACE  ----------------------
 OPTION  ==>                                                   SCROLL ==> CSR
                         L - MESSAGE LOG   J - JCL
   LIST PACKAGES    ==>                  APPEND TO LIST?    ==> N
   WHERE STATUS     ==>                  UTILITY STATUS     ==>
   WHERE DATE       ==>           THRU DATE  ==>           OLDER THAN  ==>

==============================================================================
   REQUEST  ==>              (PRINTLOG, REPORT, REPORTX, REMAKE, DELETE)

==============================================================================
   LL - LIST AUDIT LOG  LA - LIST ACTIONS   LR - LIST REGISTERED ELEMENTS

==============================================================================
     PACKAGE ID       Endevor-STATUS UTILITY-STATUS DATE      TIME    USER
 --   ----------------   -------------- -------------- --------  -----  ------
```

Figure 6.1
The Package Utility ISPF Panel

Fields available for entering search criteria are:

Field	**Usage**
Packages	Enter a 1 to 16 character package ID, leave blank or enter a wild carded package ID.
Days older than	Enter a 1 to 3 character value, range ‘1’ to ‘999’.
Status	Endevor status value.
Utility Status	Utility status value.
Date	Utility last update date. If enter with through value, considered bottom of range. If entered alone, then only packages with this date will meet criteria.
Through date	Top of date range. Must be used with first date value.

Once you have entered in field values, press –enter- and you will generate a list of package ids. Once a package is in the list, it is eligible for line commands and for report generation.

Interface Line Commands

There are several line commands available with the interface. They are as follows:

Cmd	**Usage**
LL	List all log entries for the package ID.
LA	List all action summary records for this package ID.
LR	List all registered elements for this package ID.

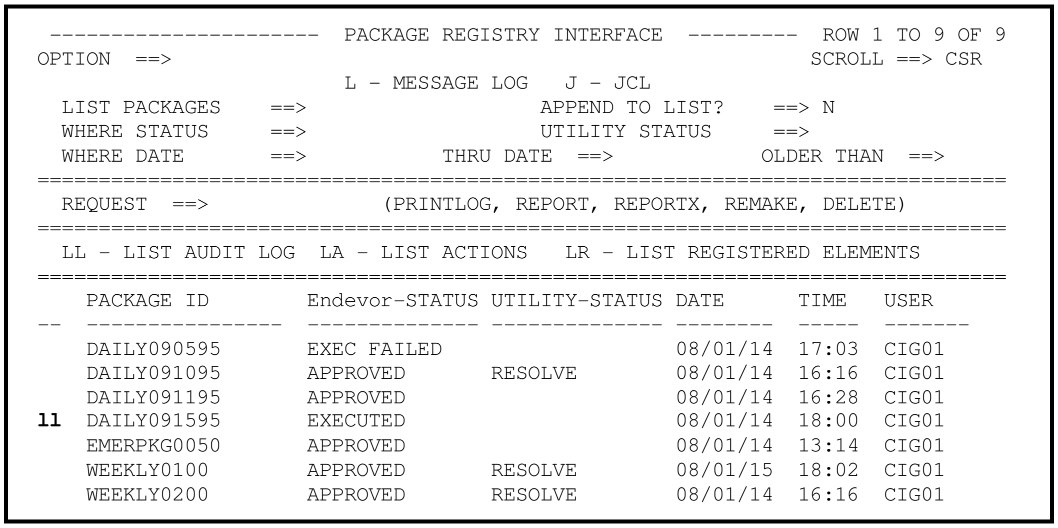

```
 ----------------------  PACKAGE REGISTRY INTERFACE  ---------  ROW 1 TO 9 OF 9
OPTION  ==>                                                    SCROLL ==> CSR
                         L - MESSAGE LOG   J - JCL
  LIST PACKAGES    ==>                 APPEND TO LIST?    ==> N
  WHERE STATUS     ==>                 UTILITY STATUS     ==>
  WHERE DATE       ==>         THRU DATE  ==>          OLDER THAN  ==>
===============================================================================
  REQUEST  ==>            (PRINTLOG, REPORT, REPORTX, REMAKE, DELETE)
===============================================================================
  LL - LIST AUDIT LOG  LA - LIST ACTIONS   LR - LIST REGISTERED ELEMENTS
===============================================================================
    PACKAGE ID        Endevor-STATUS UTILITY-STATUS DATE      TIME   USER
--  ----------------  -------------- -------------- --------  -----  -------
    DAILY090595       EXEC FAILED                   08/01/14  17:03  CIG01
    DAILY091095       APPROVED       RESOLVE        08/01/14  16:16  CIG01
    DAILY091195       APPROVED                      08/01/14  16:28  CIG01
ll  DAILY091595       EXECUTED                      08/01/14  18:00  CIG01
    EMERPKG0050       APPROVED                      08/01/14  13:14  CIG01
    WEEKLY0100        APPROVED       RESOLVE        08/01/15  18:02  CIG01
    WEEKLY0200        APPROVED       RESOLVE        08/01/14  16:16  CIG01
```

Figure 6.2
Sample of List Package Test Results

Entering '**LL**' in the line command field for package 'DAILY091095' would result in the following list. Note that the log is a sequential log and that it contains three different formats: 1) activity log messages, 2) element log messages, and 3) general messages that have been issued to the package ID.

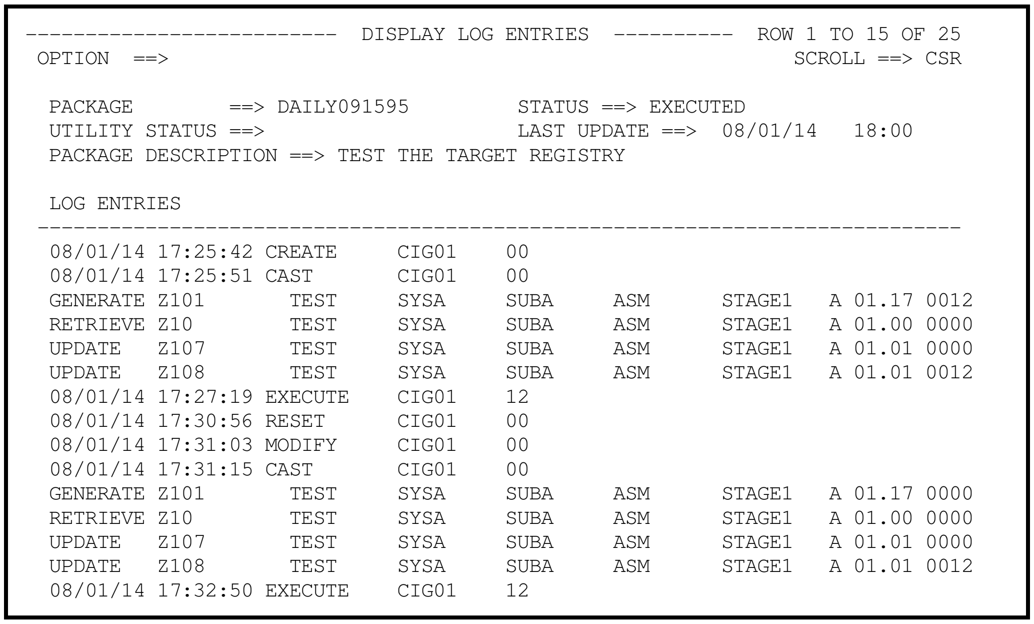
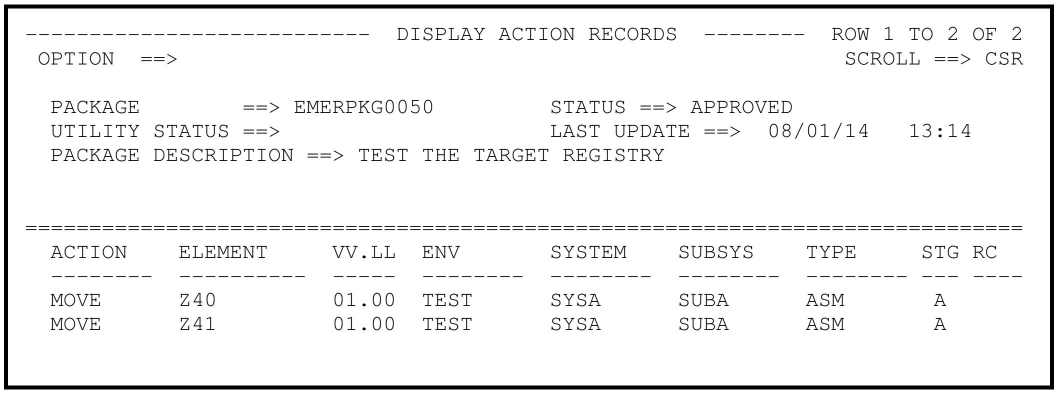

```
-------------------------  DISPLAY LOG ENTRIES  ----------  ROW 1 TO 15 OF 25
OPTION  ==>                                                   SCROLL ==> CSR

 PACKAGE        ==> DAILY091595         STATUS ==> EXECUTED
 UTILITY STATUS ==>                     LAST UPDATE ==>  08/01/14   18:00
 PACKAGE DESCRIPTION ==> TEST THE TARGET REGISTRY

 LOG ENTRIES
----------------------------------------------------------------------------
 08/01/14 17:25:42 CREATE     CIG01    00
 08/01/14 17:25:51 CAST       CIG01    00
 GENERATE Z101       TEST     SYSA     SUBA     ASM      STAGE1   A 01.17 0012
 RETRIEVE Z10        TEST     SYSA     SUBA     ASM      STAGE1   A 01.00 0000
 UPDATE   Z107       TEST     SYSA     SUBA     ASM      STAGE1   A 01.01 0000
 UPDATE   Z108       TEST     SYSA     SUBA     ASM      STAGE1   A 01.01 0012
 08/01/14 17:27:19 EXECUTE    CIG01    12
 08/01/14 17:30:56 RESET      CIG01    00
 08/01/14 17:31:03 MODIFY     CIG01    00
 08/01/14 17:31:15 CAST       CIG01    00
 GENERATE Z101       TEST     SYSA     SUBA     ASM      STAGE1   A 01.17 0000
 RETRIEVE Z10        TEST     SYSA     SUBA     ASM      STAGE1   A 01.00 0000
 UPDATE   Z107       TEST     SYSA     SUBA     ASM      STAGE1   A 01.01 0000
 UPDATE   Z108       TEST     SYSA     SUBA     ASM      STAGE1   A 01.01 0012
 08/01/14 17:32:50 EXECUTE    CIG01    12
```

Figure 6.3
Show Log Entries

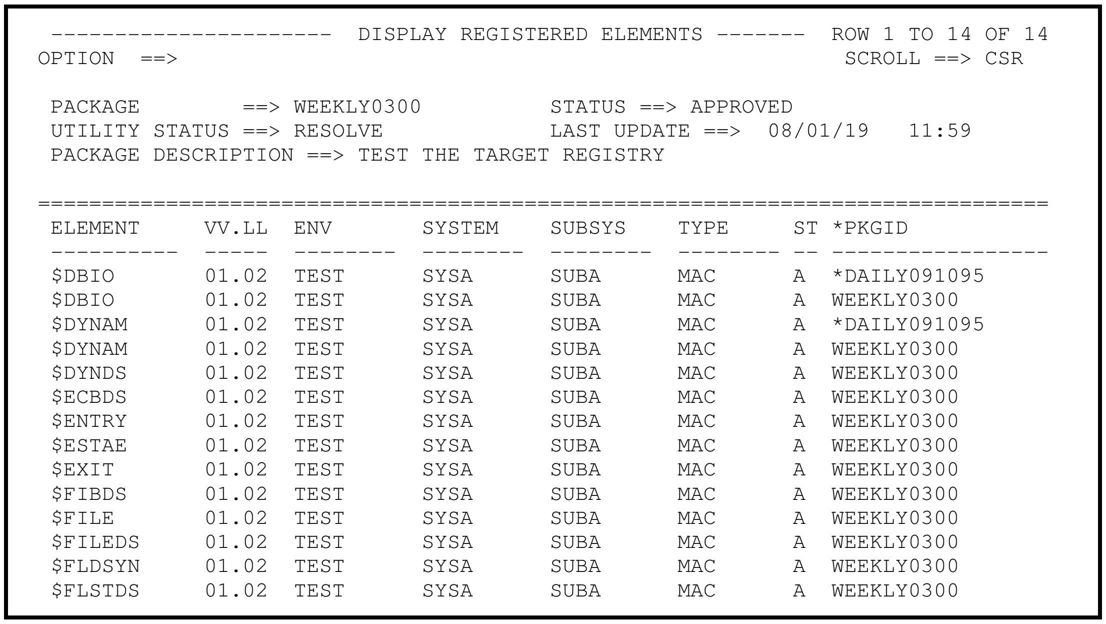
If you enter an '**LA**' in the line command field, the following panel will be displayed.



```
---------------------------  DISPLAY ACTION RECORDS  --------  ROW 1 TO 2 OF 2
 OPTION  ==>                                                   SCROLL ==> CSR

  PACKAGE        ==> EMERPKG0050         STATUS ==> APPROVED
  UTILITY STATUS ==>                     LAST UPDATE ==>  08/01/14   13:14
  PACKAGE DESCRIPTION ==> TEST THE TARGET REGISTRY


===============================================================================
  ACTION    ELEMENT     VV.LL  ENV       SYSTEM    SUBSYS    TYPE     STG RC
  --------  ----------  -----  --------  --------  --------  -------- --- ----
  MOVE      Z40         01.00  TEST      SYSA      SUBA      ASM       A
  MOVE      Z41         01.00  TEST      SYSA      SUBA      ASM       A
```

Figure 6.4
Show Action Records

Likewise, if you enter a '**LR**' in the line command field, the following panel will be displayed. Note that the display contains elements and different package ids than the current. Any elements in the list that contain a package with a '*' append in front of the ID, are collisions. This is one way to determine which packages and elements are in conflict with a given package. If you were to issue a **'LR'** line command for 'daily091095' you will see 'weekly0300' elements also listed.

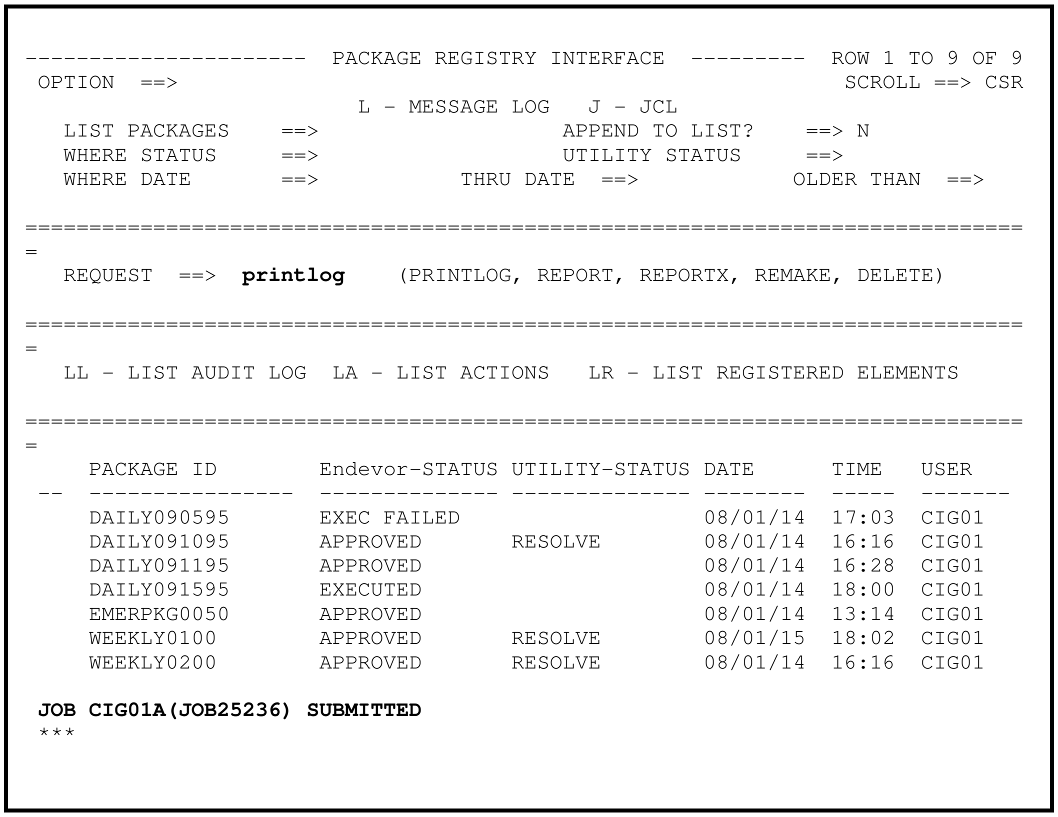


```
 ----------------------  DISPLAY REGISTERED ELEMENTS -------  ROW 1 TO 14 OF 14
OPTION  ==>                                                  SCROLL ==> CSR

 PACKAGE        ==> WEEKLY0300          STATUS ==> APPROVED
 UTILITY STATUS ==> RESOLVE             LAST UPDATE ==>  08/01/19   11:59
 PACKAGE DESCRIPTION ==> TEST THE TARGET REGISTRY

===============================================================================
 ELEMENT     VV.LL  ENV       SYSTEM    SUBSYS    TYPE     ST *PKGID
 ----------  -----  --------  --------  --------  -------- -- -----------------
 $DBIO       01.02  TEST      SYSA      SUBA      MAC      A  *DAILY091095
 $DBIO       01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
 $DYNAM      01.02  TEST      SYSA      SUBA      MAC      A  *DAILY091095
 $DYNAM      01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
 $DYNDS      01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
 $ECBDS      01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
 $ENTRY      01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
 $ESTAE      01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
 $EXIT       01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
 $FIBDS      01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
 $FILE       01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
 $FILEDS     01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
 $FLDSYN     01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
 $FLSTDS     01.02  TEST      SYSA      SUBA      MAC      A  WEEKLY0300
```

Figure 6.5
Show Registered Elements

Generating and Submitting Reports

Once there is a list of packages available, JCL can be created to submit either CIGPKUT1 command verbs or CIGPKUT2 command verbs. From this same first screen, enter the type of report or action desired. The most common used options are listed next to the 'REQUEST' field. Enter in command verb and press -enter-. Syntax will be generated and JCL will be created and submitted through file tailoring. Figure 6.6 below shows an example of submitting a PRINTLOG request. A batch job is submitted by the Package Utilities ISPF front-end.



```
----------------------  PACKAGE REGISTRY INTERFACE  ---------  ROW 1 TO 9 OF 9
 OPTION  ==>                                                   SCROLL ==> CSR
                         L - MESSAGE LOG   J - JCL
   LIST PACKAGES    ==>                   APPEND TO LIST?    ==> N
   WHERE STATUS     ==>                   UTILITY STATUS     ==>
   WHERE DATE       ==>           THRU DATE  ==>           OLDER THAN  ==>

==============================================================================
=
   REQUEST  ==>  printlog    (PRINTLOG, REPORT, REPORTX, REMAKE, DELETE)

==============================================================================
=
   LL - LIST AUDIT LOG  LA - LIST ACTIONS   LR - LIST REGISTERED ELEMENTS

==============================================================================
=
     PACKAGE ID        Endevor-STATUS UTILITY-STATUS DATE      TIME   USER
 --  ----------------  -------------- -------------- --------  -----  -------
     DAILY090595       EXEC FAILED                   08/01/14  17:03  CIG01
     DAILY091095       APPROVED       RESOLVE        08/01/14  16:16  CIG01
     DAILY091195       APPROVED                      08/01/14  16:28  CIG01
     DAILY091595       EXECUTED                      08/01/14  18:00  CIG01
     EMERPKG0050       APPROVED                      08/01/14  13:14  CIG01
     WEEKLY0100        APPROVED       RESOLVE        08/01/15  18:02  CIG01
     WEEKLY0200        APPROVED       RESOLVE        08/01/14  16:16  CIG01

 JOB CIG01A(JOB25236) SUBMITTED
 ***
```

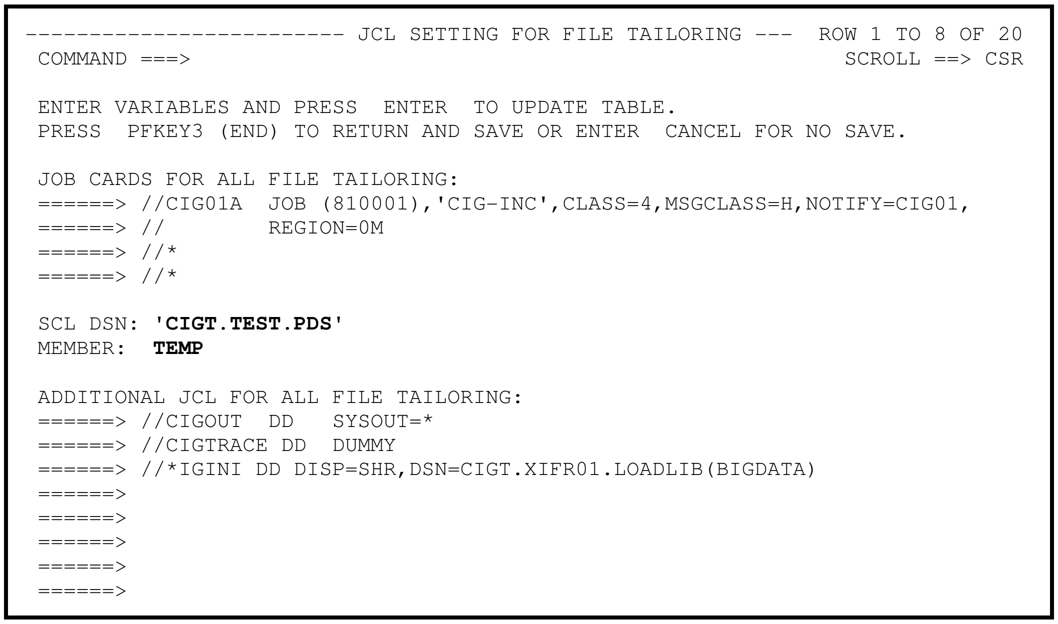
Figure 6.6
Submit a PRINTLOG Request

All the command verbs are supported through this interface. With the exception of the REMAKE and DELETE command, the system will generate one job that will process all the packages on the line. For the REMAKE and DELETE commands, the system will generate one job for each row in the table. The command verbs available are:

PRINTLOG	Generate an audit log report for all packages in list
REPORT	Generate an element by package report
REPORTX	Generate a package by element xref report
RESETID	Reset the ‘utility status’ field
CLEARLOG	Clear out the log data for packages on the list
ARCHLOG	Archive the log data for packages on the list
REMAKE	Remake each package on list that qualifies
DELETE	Issue an Endevor and Utility Delete for each package

Setting Job Card and Request Data set Values

To view your job card settings, enter a ‘J’ at the command line. This will cause the JCL management screen to appear. Note the Package Utility shares the job card and ‘additional JCL’ variable names with Endevor and FastLIST. If your configuration is using the same profile pool as either FastLIST or Endevor, these values should be set. If not, you will need to set these values the very first time. Verify that the job variables are set prior to submitting a report request. Figure 6.7 displays the JCL management screen.



```
------------------------- JCL SETTING FOR FILE TAILORING ---  ROW 1 TO 8 OF 20
 COMMAND ===>                                                   SCROLL ==> CSR

 ENTER VARIABLES AND PRESS  ENTER  TO UPDATE TABLE.
 PRESS  PFKEY3 (END) TO RETURN AND SAVE OR ENTER  CANCEL FOR NO SAVE.

 JOB CARDS FOR ALL FILE TAILORING:
 ======> //CIG01A  JOB (810001),'CIG-INC',CLASS=4,MSGCLASS=H,NOTIFY=CIG01,
 ======> //        REGION=0M
 ======> //*
 ======> //*

 SCL DSN: 'CIGT.TEST.PDS'
 MEMBER:  TEMP

 ADDITIONAL JCL FOR ALL FILE TAILORING:
 ======> //CIGOUT  DD   SYSOUT=*
 ======> //CIGTRACE DD  DUMMY
 ======> //*IGINI DD DISP=SHR,DSN=CIGT.XIFR01.LOADLIB(BIGDATA)
 ======>
 ======>
 ======>
 ======>
 ======>
```

Figure 6.7
JCL Management Panel

Also verify that the CIGSKL01 and CIGSKL02 file tailoring skeletons have been modified to reflect your site specific library names.

Tutorials

Each of the panels has a set of tutorials to explain in detail the fields and functionality of the particular panel. To access this tutorial, press PFKEY1. There may be sub tutorials available on some panels.

Chapter 6 CIGPKUT1 - REPORTS, LOG and RESETID Commands

This chapter contains:

- Instructions on how to run reports, log maintenance, and RESETID jobs.
- JCL to execute CIGPKUT1.
- Report formats.
- Error conditions.

Introduction

The Package Utilities has one general purpose utility for reporting and working with log data in the Package Utilities Registry File. This utility is CIGPKUT1 and uses the following verbs:

PRINTLOG	- Print a log report.
ARCHLOG	- Archive log data.
CLEARLOG	- Clear out log data.
REPORT	- Produce an element by package report.
REPORTX	- Produce a package by element report.
RESETID	- Clear the Utility Status Field.

CIGPKUT1 JCL To Run Reports

Figure 7.1 below shows the JCL required to run the utility. This utility can also be run in a CLIST or REXX procedure. You will find a copy of this JCL in the SAMPLIB downloaded from the tape. Change the input to reflect your report request. Note, //CIGIN can be instream data.

```
//**(JOBCARD)
//**
//* ------------------------------------------------------------------*
//* NAME:    CIGJCL51                                                 *
//* PURPOSE: JCL TO RUN CIGPKUT1 UTILITY PROGRAM                      *
//* ------------------------------------------------------------------*
//* ------------------------------------------------------------------*
//* THIS JCL IS SET UP TO INPUT SYNTAX FROM THE SAMPLIB DATASET.      *
//* MODIFY THE JCL TO INPUT USER SYNTAX. ALSO THE CIGARCH DDNAME      *
//* IS ONLY REQUIRED FOR ARCHLOG ACTION.                              *
//* ------------------------------------------------------------------*
//*  TO USE THIS JCL, YOU MUST:                                       *
//*         1) INSERT A VALID JOB CARD WITH VALID CLASS               *
//*         2) MAKE SURE THAT THE STEPLIB POINTS TO THE DATASET THAT  *
//*            CONTAINS CIGINI AND CIGFEXEC.                          *
//*         3) CHANGE FLHQ1, FLHQ2, QUAL1 AND QUAL2 AS PER YOUR       *
//*            INSTALLATION SHEET                                     *
//* ------------------------------------------------------------------*
//*
//* JCL TO RUN THE FOLLOWING BATCH COMMAND VERBS AGAINST              *
//* THE PACKAGE UTILITY REGISTRY FILE.                                *
//*    - PRINTLOG PACKAGE 'XX' .                                      *
//*    - CLEARLOG PACKAGE 'XX' .                                      *
//*    - ARCHLOG  PACKAGE 'XX' .                                      *
//*    - REPORT   PACKAGE 'XX' .                                      *
//*    - REPORTX  PACKAGE 'XX' .                                      *
//* ---------------------------------------------------------------- *
//STEP1    EXEC PGM=CIGPKUT1
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGLOG   DD  SYSOUT=*
//CIGRPT   DD  SYSOUT=*
//CIGARCH  DD  DSN=FLHQ1.ARCHDATA,DISP=SHR <= ONLY FOR ARCHLOG
//CIGIN    DD  DSN=FLHQ1.FLHQ1.SAMPLIB(PKGIVP1),DISP=SHR
```

Figure 7.1
JCL to Execute CIGPKUT1

CIGPKUT1 REPORT Syntax

Figure 7.2 contains all syntax supported by the CIGPKUT1 utility.

```
PRINTLOG | CLEARLOG | ARCHLOG  | REPORT    |REPORTX       | RESETID

      PACKAGE        'package-name'

      WHERE STATUS EQUALS 'ndvr-pkg-status'

      WHERE UTILITY STATUS EQUALS 'utility-status'

      WHERE FROM DATE EQUAL 'YY/MM/DD' [THROUGH DATE EQUAL 'YY/MM/DD']

      WHERE OLDER THAN '999' DAYS
.
```

Figure 7.2
CIGPKUT1 Syntax

'Where' Clauses

Note that there are six command verbs. Each one can optionally use the 'where' clauses to further limit the output produced. The minimum required is a verb and a package ID. For example:

```
PRINTLOG PACKAGE 'TESTXX'  .
```

Where the package is a one to 16 valid package ID value, enclosed in quotes. All verbs use the same required format.

The 'where' clause allows you to further limit or select packages to be included in the reports. The rules for the where clauses are as follows:

WHERE STATUS EQUALS *status*

This clause will limit the report to packages that meet the stated Endevor status value.

WHERE UTILITY STATUS EQUALS *status*

This clause will limit the verb to packages that meet the stated Package Utility status value. Valid values are:

RESOLVE	packages that have been in a collision
REMAKE	packages that are in the process of a REMAKE
IN-DELETION	packages that are in the process of being deleted from the registry and Endevor

WHERE FROM DATE EQUALS ‘yy/mm/dd’ [THROUGH DATE EQUALS ‘yy/mm/dd’] .

If only one date parameter is sent, then the search will return packages whose utility last update date is equal to the value. If both dates are returned, then the search will return packages whose utility last update date is within the range of dates, inclusive. The date range must be valid to pass parsing requirements.

WHERE OLDER THAN ‘999’ DAYS.

This clause is mutually exclusive with date ranges. You provide a one- to three-character value from ‘1’ to ‘999’. The search will return packages whose utility last update date is less than today’s date minus the ‘older than’ value.

CIGPKUT1 Execution Summary

Figure 7.3 contains an example of the report execution summary. Note this information is written to the CIGLOG ddname.

```
15:29:10  FST0281I ----- COMMON INITIALIZATION INFORMATION -----
15:29:10  FST0251I PRODUCT LOAD LIBRARY......... CIGT.XIFR01.LOADLIB
15:29:10  FST0280I ALTERNATE CIGINI ALLOWED?.... N
15:29:10  FST0269I PACKAGE VSAM FILE........... CIGT.PACKAGE.DB

DATE 08/01/13 TIME 15:29:12     P A C K A G E    U T I L I T Y, RELEASE 12.0
                                 E X E C U T I O N  R E P O R T
15:29:12  FST1102I  PARSER BEGINS
15:29:12  FST0020I  PRINTLOG PACKAGE 'TEST*' .
15:29:12  FST0020I  REPORT   PACKAGE 'TEST*' .
15:29:12  FST1103I  PARSER ENDS, RC=0000
15:29:14  PKG3163I  PRINTLOG COMPLETED SUCCESSFULLY FOR PACKAGE 'TEST1' .
15:29:14  PKG3163I  PRINTLOG COMPLETED SUCCESSFULLY FOR PACKAGE 'TEST11' .
15:29:14  PKG3163I  PRINTLOG COMPLETED SUCCESSFULLY FOR PACKAGE 'TEST2' .
15:29:14  PKG3163I  PRINTLOG COMPLETED SUCCESSFULLY FOR PACKAGE 'TEST22' .
15:29:14  PKG3163I  PRINTLOG COMPLETED SUCCESSFULLY FOR PACKAGE 'TEST3' .
15:29:14  PKG3163I  PRINTLOG COMPLETED SUCCESSFULLY FOR PACKAGE 'TEST33' .
15:29:14  PKG3163I  PRINTLOG COMPLETED SUCCESSFULLY FOR PACKAGE 'TEST4' .
```

Figure 7.3
CIGPKUT1 Execution Log

PRINTLOG Command The Audit Log Report

Figure 7.4 contains an example of the PRINTLOG syntax.

```
PRINTLOG PACKAGE ‘WEEKLY0500’  .
```

Figure 7.4
Printlog Syntax Example

Figure 7.5 contains the PRINTLOG report format. Note this information is written to the CIGRPT ddname. This report lists the package log entries in the order that they occurred.

```
DATE 08/01/13 TIME 15:29:14   P A C K A G E  U T I L I T Y, RELEASE 12.0   PAGE    1
                               A U D I T  L O G   R E P O R T

FOR PACKAGE: WEEKLY0500  STATUS: APPROVED  UTILSTAT:  UPDATE: 08/01/13  13:07  USER: CIG01A

     ACTIVITY        USER        DATE       TIME     RC
     CREATE          CIG01A      08/01/13   12:57    00
     CAST            CIG01A      08/01/13   12:57    00
     MOVE       $CPOOL      (01.00)   TEST     SYSA      SUBA      MAC A RC=(0000)
     EXECUTE         CIG01A      08/01/13   13:01    04
     REMAKE          JTPSXXX     08/01/13   13:01    00
     COMMIT          JTPSXXX     08/01/13   13:02    00
     DELETE          JTPSXXX     08/01/13   13:02    00
     CREATE          JTPSXXX     08/01/13   13:02    00
     CAST            JTPSXXX     08/01/13   13:02    00
     MOVE       $CPOOL      (01.00)   TEST     SYSA      SUBA      MAC  B RC=(0000)
     EXECUTE         CIG01A      08/01/13   13:06    08
     REMAKE          JTPSXXX     08/01/13   13:06    00
     COMMIT          CIG01A      08/01/13   13:06    00
     CREATE          CIG01A      08/01/13   13:07    00
     CAST            CIG01A      08/01/13   13:07    00
```

Figure 7.5
PRINTLOG Report

There are three types of log entries. The first type is the activity type log entry. This is the formatted type of entry. For example:

```
       CREATE          CIG01A      08/01/13   13:07    00
        CAST             CIG01A      08/01/13   13:07    00
```

The package activity, user, activity date and time, and return code are entered.

The second type of log entry is the element type entry. For every element action performed in the package, there will be an entry written to the log. For example:

```
MOVE       $CPOOL      (01.00)   TEST     SYSA      SUBA      MAC  B RC=(0000
```

The third type of log entry is a message type of entry. All warning, error, and informational messages are written to the log. For example:

```
PACKAGE SET TO 'RESOLVE' STATUS DUE TO ELEMENT COLLISION(S).
** COLLISION WITH PACKAGE TEST33 -  ELEMENT  $DBIO/PROD/SYSA/SUBA/ASM/1
```

Other important fields:

Status	Endevor status value
Utility Status	Package Utility status value
Update	Utility last update date and time
User	Utility last update user

REPORT Command The Element by Package Report

Figure 7.6 contains an example of the REPORT syntax.

```
REPORT PACKAGE ‘WEEKLY0300’   .
```

Figure 7.6
Example of the REPORT Syntax

Figure 7.7 contains the REPORT report format. Note this information is written to the CIGRPT ddname. This report lists elements per package and will also list all elements that are registered to another package ID.

```
DATE 08/01/13 TIME 15:44:51     P A C K A G E   U T I L I T Y, RELEASE 12.0   PAGE    1

                      E L E M E N T S  B Y  P A C K A G E  R E P O R T

     ACTION     ELEMENT        VV.LL   ENV         SYSTEM       SUBSYS       TYPE        STG

FOR PACKAGE: WEEKLY0300  STATUS: APPROVED  UTILSTAT: RESOLVE  UPDATE: 08/01/19 11:59 USER: CIG01

     MOVE       $DBIO          01.00   TEST        SYSA         SUBA         MAC          A
     ** ALSO REGISTERED IN PACKAGE ID: DAILY091095
     MOVE       $DYNAM         01.00   TEST        SYSA         SUBA         MAC          A
     ** ALSO REGISTERED IN PACKAGE ID: DAILY091095
     MOVE       $DYNDS         01.00   TEST        SYSA         SUBA         MAC          A
     MOVE       $ECBDS         01.00   TEST        SYSA         SUBA         MAC          A
     MOVE       $ENTRY         01.00   TEST        SYSA         SUBA         MAC          A
     MOVE       $ESTAE         01.01   TEST        SYSA         SUBA         MAC          A
     MOVE       $EXIT          01.01   TEST        SYSA         SUBA         MAC          A
     MOVE       $FIBDS         01.01   TEST        SYSA         SUBA         MAC          A
     MOVE       $FILE          01.01   TEST        SYSA         SUBA         MAC          A
     MOVE       $FILEDS        01.01   TEST        SYSA         SUBA         MAC          A
     MOVE       $FLDSYN        01.00   TEST        SYSA         SUBA         MAC          A
     MOVE       $FLSTDS        01.02   TEST        SYSA         SUBA         MAC          A
```

Figure 7.7
REPORT Report

For each package that qualifies in the search, the report will list all actions and elements that are included in the package. This information comes from the action summary records, which are available from **CAST** time until deletion. Per element, all ‘registered’ elements will also be listed. A ‘registered’ element is an element that has been **CAST** but not yet **EXECUTED.**

Other important fields:

Status	Endevor status value
Utility Status	Package Utility status value
Update	Utility last update date and time
User	Utility last update user

REPORTX The Package by Element Report

Figure 7.8 contains an example of the REPORTX syntax.

```
REPORTX PACKAGE '*'  .
```

Figure 7.8
Example of the REPORTX Syntax

Figure 7.9 contains the REPORTX report format. Note this information is written to the CIGRPT ddname. This report lists every package in which an element is contained.

```
P A C K A G E S   B Y   E L E M E N T S   R E P O R T

    PACKAGE          STATUS       UTILSTAT     USER     DATE     TIME

FOR ELEMENT: $DBIO      (01.00) TEST/SYSA/SUBA/MAC/A
    DAILY091095      APPROVED     RESOLVE      CIG01    08/01/14 16:16
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59
FOR ELEMENT: $DYNAM     (01.00) TEST/SYSA/SUBA/MAC/A
    DAILY091095      APPROVED     RESOLVE      CIG01    08/01/14 16:16
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59

FOR ELEMENT: $DYNDS     (01.00) TEST/SYSA/SUBA/MAC/A
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59

FOR ELEMENT: $ECBDS     (01.00) TEST/SYSA/SUBA/MAC/A
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59

FOR ELEMENT: $ENTRY     (01.00) TEST/SYSA/SUBA/MAC/A
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59

FOR ELEMENT: $ESTAE     (01.01) TEST/SYSA/SUBA/MAC/A
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59

FOR ELEMENT: $EXIT      (01.01) TEST/SYSA/SUBA/MAC/A
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59

FOR ELEMENT: $FIBDS     (01.01) TEST/SYSA/SUBA/MAC/A
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59

FOR ELEMENT: $FILE      (01.01) TEST/SYSA/SUBA/MAC/A
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59

FOR ELEMENT: $FILEDS    (01.01) TEST/SYSA/SUBA/MAC/A
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59

FOR ELEMENT: $FLDSYN    (01.00) TEST/SYSA/SUBA/MAC/A
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59

FOR ELEMENT: $FLSTDS    (01.02) TEST/SYSA/SUBA/MAC/A
    WEEKLY0300       APPROVED     RESOLVE      CIG01    08/01/19 11:59

FOR ELEMENT: ELM0X0     (01.01) TEST/SYSA/SUBA/ASM/A
    WEEKLY0100       APPROVED     RESOLVE      CIG01    08/01/15 18:02
```

```
FOR ELEMENT: ELM0X1     (01.01) TEST/SYSA/SUBA/ASM/A
    WEEKLY0100       APPROVED     RESOLVE      CIG01    08/01/15 18:02

FOR ELEMENT: ELM0X10    (01.01) TEST/SYSA/SUBA/ASM/A
    WEEKLY0100       APPROVED     RESOLVE      CIG01    08/01/15 18:02
    WEEKLY0200       APPROVED     RESOLVE      CIG01    08/01/14 16:16

FOR ELEMENT: ELM0X11    (01.01) TEST/SYSA/SUBA/ASM/A
    WEEKLY0100       APPROVED     RESOLVE      CIG01    08/01/15 18:02
    WEEKLY0200       APPROVED     RESOLVE      CIG01    08/01/14 16:16

FOR ELEMENT: ELM0X12    (01.00) TEST/SYSA/SUBA/ASM/A
    WEEKLY0200       APPROVED     RESOLVE      CIG01    08/01/14 16:16

FOR ELEMENT: ELM0X13    (01.00) TEST/SYSA/SUBA/ASM/A
    WEEKLY0200       APPROVED     RESOLVE      CIG01    08/01/14 16:16

FOR ELEMENT: Z10        (01.00) TEST/SYSA/SUBA/ASM/A
    DAILY091595      EXECUTED                  CIG01    08/01/14 18:00
```

Figure 7.9
REPORTX Report

For each action summary found, the report will list all package that contain the element. This information comes from the action summary records, which are available from **CAST** time.

Other important fields:

Status	Endevor status value
Utility Status	Package Utility status value
Date/Time	Utility last update date and time
User	Utility last update user

ARCHLOG Command Archiving log records

Unlike the report command verbs, the output from ARCHLOG command is not a report, but an execution summary and also stores the data into a data set. To be able to run the ARCHLOG command, you must include a CIGARCH ddname in the JCL. If this file is omitted the job will end with a message and a return code of 12. The attributes for the CIGARCH data set are as follows:

LRECL	**255**
RECFM	**FB**
BLOCKSIZE	**27795**
DSORG	**PS**

The file layouts for the ARCHLOG data can be found in APPENDIX C. of this manual.

Figure 7.10 contains an example of ARCHLOG syntax.

```
ARCHLOG PACKAGE ‘WEEKLY0300’ .
```

Figure 7.10
Example of the ARCHLOG Syntax

Figure 7.11 contains and example of the execution summary you will see after an ARCHLOG.

```
15:29:10  FST0281I ----- COMMON INITIALIZATION INFORMATION -----
15:29:10  FST0251I PRODUCT LOAD LIBRARY......... CIGT.XIFR01.LOADLIB
15:29:10  FST0280I ALTERNATE CIGINI ALLOWED?.... N
15:29:10  FST0269I PACKAGE VSAM FILE............ CIGT.PACKAGE.DB

DATE 08/01/13 TIME 15:29:12     P A C K A G E    U T I L I T Y, RELEASE 12.0
                                 E X E C U T I O N  R E P O R T
15:29:12  FST1102I  PARSER BEGINS
15:29:12  FST0020I  ARCHLOG PACKAGE ‘WEEKLY0300' .
15:29:12  FST1103I  PARSER ENDS, RC=0000

15:29:14  PKG3163I  ARCHLOG COMPLETED SUCCESSFULLY FOR PACKAGE 'WEEKLY0300' .
```

Figure 7.11
Output from the ARCHLOG Command Execution

CLEARLOG Deleting Records from the Log

Unlike the report command verbs, the output from CLEARLOG command is not a report, but an execution summary. Figure 7.12 contains an example of CLEARLOG syntax.

```
CLEARLOG PACKAGE ‘WEEKLY0300’ .
```

Figure 7.12
Example of CLEARLOGD Syntax

Figure 7.13 contains and example of the execution summary you will see after an CLEARLOG.

```
15:29:10  FST0281I ----- COMMON INITIALIZATION INFORMATION -----
15:29:10  FST0251I PRODUCT LOAD LIBRARY......... CIGT.XIFR01.LOADLIB
15:29:10  FST0280I ALTERNATE CIGINI ALLOWED?.... N
15:29:10  FST0269I PACKAGE VSAM FILE........... CIGT.PACKAGE.DB

DATE 08/01/13 TIME 15:29:12     P A C K A G E    U T I L I T Y, RELEASE 12.0
                                 E X E C U T I O N  R E P O R T
15:29:12  FST1102I  PARSER BEGINS
15:29:12  FST0020I  CLEARLOG PACKAGE ‘WEEKLY0300' .
15:29:12  FST1103I  PARSER ENDS, RC=0000

15:29:14  PKG3163I  CLEARLOG COMPLETED SUCCESSFULLY FOR PACKAGE 'WEEKLY0300' .
```

Figure 7.13
Output from the CLEARLOG Command Execution

DISABLE DELETE Option And Clearlog

If Package Utilities has been implemented with the DISABLE DELETE option set in the CIGINI, users will not be able to perform a Package Utilities Clearlog function. If they attempt to perform a Clearlog they will get the following message in the CIGLOG.

```
PKG3304E  DELETE OR CLEARLOG REQUEST DENIED.
PKG3305E  BOTH FUNCTIONS ARE DISABLED FOR YOUR INSTALLATION.
```

Fig 5.14 Disable Delete Messages

RESETID Command   Clearing the Package Utility field

The RESETID command verb is different from the other CIGPKUT1 verbs in that it modifies the contents of package registry records. Upon a failed REMAKE or a RESOLVE condition, you can perform a RESETID verb against the package ID and clear the Utility Status field. This would be done in the case of a RESOLVE condition that has been fixed by performing a reset, modify and re-cast of one of the packages involved in the collision. If the rest of the packages do not need to be modified, issuing a RESETID command will clear the Utility Field, making the package eligible for approval and execution. The execution of this command is logged in the audit trail.

Figure 7.14 contains an example of the RESETID syntax.

```
RESETID PACKAGE '*'
      WHERE UTILITY STATUS EQUAL 'RESOLVE' .
```

Figure 7.14
Example of RESETID Syntax

Using this syntax, all packages with the Utility Status of RESOLVE would be cleared.

CIGPKUT1	Return Code	Meaning
	00	All processes completed successfully
	12	All error conditions. Parser, missing file, etc. Check log for messages.

All messages sent from the utility will begin with the prefix FST or PKG and will be written out to CIGLOG.

CIG Package Utilities

Appendix A: File Layout for REMAKE User Program

The following is the file layout of the user block passed in register one to user program prior to performing the REMAKE action:

```
      MACRO
&NAME    $USRDS &DSCT=YES
         AIF   ('&DSCT' NE 'YES').SKPDSCT
$USRDS DSECT
         AGO   .SKPEQU
.SKPDSCT ANOP
$USRDS DS      0F
.SKPEQU  ANOP
**********************************************************************
*                                                                    *
*         $USRDS - BLOCK FOR USRPS TO PROVIDE THE PACKAGE            *
*                    REGISTRY OVERRIDE PACKAGE DEFINE DATA.          *
*                                                                    *
**********************************************************************
USRPID    DC    CL4'USRP'          BLOCK ID
USRPPKID  DC    CL16' '            CURRENT PACKAGE ID
USRPPNEW  DC    CL16' '            NEW PACKAGE ID
USRPCOMM  DC    CL50' '            PACKAGE COMMENT
USRPTYPE  DC    CL10' '            PACKAGE TYPE     (STANDARD )
USRPBOEN DC     CL1' '             BACKOUT ENABLED FOR PACKAGE
USRP$BYS  EQU    C'Y'                   YES
USRP$BNO  EQU   C'N'                    NO
USRPPSHR  DC    CL1' '             SHARE   ENABLED FOR PACKAGE
USRP$SYS  EQU   C'Y'                    YES
USRP$SNO  EQU   C'N'                    NO          (NO)
USRPPAPD  C    CL1' '              APPEND PACKAGE ?
USRP$PYS  EQU   C'Y'                    YES
USRP$PNO  EQU   C'N'                    NO          (NO)
USRPEWSD  DC    CL7' '             EXECUTION WINDOW START DATE
USRPEWST  DC    CL5' '             EXECUTION WINDOW START TIME
USRPEWED  DC    CL7' '             EXECUTION WINDOW END DATE
USRPEWET  DC    CL5' '             EXECUTION WINDOW END TIME
USRSCLDD  DC    CL8' '             IMPORT SCL DD. DEFAULT IS CIGSCL01
USRRC     DC    F'0'               USER EXIT RETURN CODE
USROK     EQU   0                  CONTINUE WITH REMAKE
USRWARN   EQU   4                  FAILING WITH RC=4
USRFAIL   EQU   8                  FAIL WITH RC=8
USRSEVER  EQU   12                 FAIL WITH RC=12  (SEVERE ERROR)
USRDCURR  DC    CL1' '             COMMIT AND DELETE PKG IF NEW NAME
USRP$CYS  EQU   C'Y'                    YES
USRP$CNO  EQU   C'N'                    NO          (NO)
USRPMSGS  DC    CL132' '           MESSAGE FIELD
USRP_LEN  EQU   *-$USRDS           LENGTH OF $USRDS BLOCK
         MEND
```

CIG Package Utilities

Appendix B: Example - REMAKE Program

This program can be found in the Samplib downloaded from the installation tape, member name TESTPGM. Modify the program to meet your customization requirements.

```
TESTPGM TITLE 'SAMPLE PROGRAM FOR PASSING $USRDS'
***********************************************************************
*   DESCRIPTION: THIS IS AN EXAMPLE OF A PROGRAM THAT CAN BE          *
*                BE CALLED TO OVERRIDE THE VALUES IN THE              *
*                $USRDS OR TO CANCEL THE REMAKE OF THE PACKAGE.       *
*                                                                     *
*   REGISTERS ON ENTRY:                                               *
*                                                                     *
*                0(R1) --> $USRDS      PACKAGE UTILITY EXIT BLOCK     *
*                                                                     *
*   REGISTER USAGE:                                                   *
*                                                                     *
*                R9     -> $USRDS                                     *
*                R12    -> BASE PROGRAM                               *
*                R13    -> STACK USED FOR STANDARD IBM USAGE          *
*                                                                     *
*   USAGE NOTES:  THIS PROGRAM WILL BE CALLED PRIOR TO PERFORMING     *
*                 A REMAKE ACTION THROUGH THE PACKAGE UTILITY.        *
*                 AT THIS POINT THE USER CAN OVERRIDE ANY OF THE      *
*                 VALUES IN THE BLOCK AS WELL AS CANCEL THE REMAKE    *
*                 BY RETURNING  A NON-ZERO RETURN CODE.               *
*                 SOME THINGS TO NOTE:                                 *
*               1 THE DEFAULT IMPORT DD FOR ACTION SCL IS CIGSCL01.   *
*                 IF THE USER CHANGES THIS VALUE, THEN THE USER MUST  *
*                 BUILD THE SCL AND INCLUDE THE DD IN THE JCL.        *
*               2.IF THE USER PROVIDES A NEW NAME, THE UTILITY WILL   *
*                 NOT DELETE AND COMMIT THE OLD ID UNLESS THE USRDCURR*
*                 FIELD IS SET TO ‘Y’.  USER WILL HAVE TO MODIFY      *
*                 STANDARD JCL TO INCLUDE A CIGSCL03 DDNAME IN THIS   *
*                 CASE. SEE CHAPTER 5, FOR MORE INFO ON THIS TOPIC.   *
*               3.IF THE USER PROVIDES EXECUTION WINDOWS THEN THEY    *
*                 MUST BE IN Endevor FORMAT.  DDMMMYY.                *
*****|***************************************************************
*********************************************************************
*   PACKAGE UTILITY EXIT BLOCK                                        *
*********************************************************************
         $USRDS
**********************************************************************
*                                                                    *
*         $USRDS - BLOCK FOR USERS TO PROVIDE THE PACKAGE            *
*                    REGISTRY OVERRIDE PACKAGE DEFINE DATA.          *
*                                                                    *
**********************************************************************
*********************************************************************
*   REQISTER EQUATES
***********************************************************************
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
```

```
**********************************************************************
*   THIS PROGRAM'S WORKAREA MAP                                      *
**********************************************************************
WORKAREA DSECT
*
SAVEAREA DS    18F
WORKLN   EQU   *-WORKAREA
         TITLE 'TESTPGM: PACKAGE UTILITY'
**********************************************************************
*   MAINLINE LOGIC                                                   *
**********************************************************************
TESTPGM CSECT
         SAVE  (14,12),,'PACKAGE UTILITY'   SAVE CALLERS REG 12(13)
         LR    R12,R15                      POINT TO THIS PROGRAM
         USING TESTPGM,R12
         L     R8,0(,R1)                    POINT TO THE $USRDS
         USING $USRDS,R8
**********************************************************************
*   GET STORAGE FOR SAVEAREA AND $NOTIFY                             *
**********************************************************************
         L     R0,=A(WORKLN)                GET SIZE OF W.A
         GETMAIN R,LV=(0)                   GET WORKING STORAGE
         ST    R1,8(R13)                    STORE NEW STACK +8(OLD)
         ST    R13,4(R1)                    STORE OLD STACK +4(NEW)
         LR    R13,R1                       POINT R13 TO OUR STACK
**********************************************************************
*   MOVE IN TEST MESSAGE                                             *
**********************************************************************
         MVC   USRPMSGS(TESTMSLN),TESTMSG   MOVE IN TEST MESSAGE
         B     MAIN9000
**********************************************************************
*   INSERT LOGIC HERE.                                               *
**********************************************************************
MAIN8000 DS    0H
         LA    R15,4                       TO CANCEL REMAKE
         ST    R15,USRRC                   SET IN USER BLOCK
         B     MAINEXIT
MAIN9000 DS    0H
         XC    USRRC,USRRC                  CLEAR RETURN CODE
MAINEXIT DS    0H
         LR    R5,R13                        SAVE NEW STACK POINTER
         L     R13,4(R13)                    POINT TO OLD STACK
**********************************************************************
*   CLEAN UP THIS PROGRAM'S STORAGE                                  *
*   NOTE: THIS HAS TO BE DONE BEFORE THE "LOAD MULTIPLE" IS          *
*   DONE BECAUSE YOU LOSE THE POINTER TO YOUR STORAGE                *
**********************************************************************
*
         L     R0,=A(WORKLN)                GET SIZE
         FREEMAIN R,A=(5),LV=(0)            FREE STORAGE
*
MAINRTRN DS    0H
         RETURN (14,12)
*
BLANKS   DC    CL80' '
TESTMSG  DC    C'THIS IS A MESSAGE TO TEST THE MESSAGE FACILITY'
TESTMSLN EQU   *-TESTMSG
         END
```

CIG Package Utilities

Appendix C: PKG@LOG Record Layout

The following file layout is for the output from the ARCHLOG action verb. The ARCHLOG verb will format and build the following records as per your request with the CIGPKUT1 utility.

This member can be found in the Samplib from the installation tape, member name PKG@LOG.

```
         MACRO
         PKG@LOG
***********************************************************************
*   THIS IS THE EXTERNAL FILE LAYOUT FOR THE ARCHIVE LOG FUNCTION.    *
***********************************************************************
PKG@LOG  DSECT                         ARCHIVE LOG OUTPUT
*
LOGEYE   DS    CL2                     RECORD ID = C'15'
LOGEYEC   EQU   C'15'                   THIS IS A 15 RECORD
LOG_ARCH_DATE   DS   CL8               DATE OF ARCHIVE REQUEST
LOG_ARCH_TIME   DS   CL5               TIME OF ARCHIVE REQUEST
LOGHEAD  DS    0H
LOGPKGID DS    CL16                    PACKAGE ID
LOGESTAT DS    CL12                    Endevor STATUS
LOGUSTAT DS    CL12                    UTILITY STATUS
LOGPTYPE DS    CL10                    PACKAGE TYPE
LOGUPDTE DS    CL8                     LAST UPDATE DATE
LOGUPTME DS    CL5                     LAST UPDATE TIME
LOGUPUSR DS    CL8                     LAST UPDATE USER
LOGETYPE DS    CL4                     LOG ENTRY TYPE
LOG_MSGS EQU   C'MSGS'                 GENERAL MESSAGE - DEFAULT
LOG_ACTD EQU   C'ACTD'                 ACTIVITY DETAIL
LOG_ELEM EQU   C'ELEM'                 ELEMENT ACTION DETAIL
LOGTSDTE DS    CL8                     LOG ENTRY DATE
LOGTSTME DS    CL5                     LOG ENTRY TIME
LOGDATA  DS    0H                      TYPE = 'MSGS'
LOGENTRY DS    CL80                    DEFAULT VARIOUS MESSAGE AREA
*
         ORG   LOGDATA                 TYPE = 'ACTD'
LOGAVTN  DS    CL10                    ACTIVITY NAME
LOGACTU  DS    CL8                     ACTIVITY USER
LOGACTD  DS    CL8                     ACTIVITY DATE
LOGACTT  DS    CL5                     ACTIVITY TIME
LOGACTRC DS    CL4                     ACTIVITY RETURN CODE
*
         ORG   LOGDATA                 TYPE = 'ELEM'
LOGACTN  DS    CL8                     ACTION NAME
LOGELM   DS    CL10                    ELEMENT
LOGENV   DS    CL8                     ENVIRONMENT
LOGSYS   DS    CL8                     SYSTEM
LOGSBS   DS    CL8                     SUBSYSTEM
LOGTYP   DS    CL8                     TYPE
LOGSTG#  DS    CL1                     STAGE NUMBER
LOGVV    DS    CL2                     VERSION
LOGDOT   DS    CL1                      DOT
LOGLL    DS    CL2                     LEVEL
         ORG   ,
*
LOGDSLN   EQU   *-PKG@LOG               DSECT SIZE
         MEND
```

CIG Package Utilities

Additional Technical Considerations

Traces Available With Package Utilities and FastLIST

CIG TRACES

The following table outlines the CIG traces available to the user and to technical support. All of the traces are enabled via allocation of a ddname in batch or with the TSO ALLOCATE function.

If in Batch ➔ //CIGLOG DD SYSOUT=*
If in TSO ➔ TSO ALLOC FI(CIGLOG) DSN(*) SHR REUSE

Trace	Purpose	Where invoked	How to allocate
CIGLOG	General log information	Available in exits and Admin utilities.	Batch or TSO
CIGVTRAX	Logical VSAM trace	Available in all functions that access the database: Exits, Admin Utilities, and the Server.	Batch or TSO
CIGZTRAX	Physical VSAM trace Lowest level	Available in all functions that access the database, Exits, Admin Utilities, and the Server.	Batch or TSO
CIGPTRAX	Parser Trace	Available in all functions that process CIG syntax directly or indirectly. Admin Utilities and the Server.	Batch or TSO
CIGCTRAX	Low level storage trace for tracing getmain/freemain activity.	Available in functions that use CIG low level stack management. Exit assembler programs, Admin Utilities, and the Server.	Batch or TSO
CIGFTP	Trace of Applet request and server response on a user level.	Available only in the Server.	Batch Only
CIGXLSTN	Trace of Server Listener Task Activity	Available only in the Server	Batch Only
CIGXSUBT	Trace of Server Sub Task Activity	Available only in the Server	Batch Only

Note that any trace will affect performance. Use these traces for informational or debug purposes.

PRINTINI Utility

PRINTINI is a program designed to help you verify the active CIGINI in the path. To use this utility, code a small CLIST as follows. The dataset called should contain the current CIGFEXEC and CIGINI file. Copy this CLIST into a SYSPROC dataset or execute from option 6.

PRINTINI Utility JCL

```
//STEP1    EXEC  PGM=PRINTINI
//STEPLIB    DD  DSN=flhq1.flhq2.LOADLIB,DISP=SHR
//CIGRPINT  DD  SYSOUT=*
```

PRINTINI Utility CLIST

```
ALLOCATE FI(CIGPRINT) DSN(*) SHR REUSE
CALL ‘flhq1.flhq2.LOADLIB(PRINTINI)'
WRITE ****SUCCESS****
FREE FI(CIGPRINT)
```

Enter “TSO PRINTINI”.

PRINTINI output example

```
---------------
 |   COMMON    |
 ---------------
 EYECATCHER..... CIG1
 LOAD LIBRARY... CIGT.ENDPROD.LOADLIB1
 WORK UNIT...... WORK
 VIO UNIT....... WORK
 ALT INI ALLOWED Y
 CONLIB......... SYS2.CONLIB
 --------------- ---
 ALT CIGINI      NO
 --------------- ---
 ---------------
 |  FASTLIST   |
 ---------------
 PASSWORD....... 9999999
 PRIMARY VSAM... CIGT.FLSTPROD
 ALTERNATE VSAM. CIGT.FLSTPROD.PATH
 COLLECT COMPS?. Y
 COLLECT CCIDS?. Y
 FG EXEC?....... Y
 FILTERS........ NONE
 ---------------
 |   PACKAGE   |
 ---------------
 PASSWORD....... 9999999
 VSAM FILE...... CIGT.PACKAGE.DB
 CAST ELEMENT... RESOLVE
 AUTO REMAKE?... Y
 LOG RECORDING?. Y
 REMAKE EMG PKG? N
 USER PGM....... TESTPGM
 ACTION=
    MOVE........ M
    GENERATE.... A
    TRANSFER.... A
    ADD......... A
    UPDATE...... A
    RETRIEVE.... A
    DELETE...... A
    PRINT....... A
    LIST........ A
    ARCHIVE..... A
    RESTORE..... A
 FILTERS........ (NONE)
 ****SUCCESS****
```

Reserved ddnames and Considerations

Reserved ddnames

Throughout this manual there will be several examples of JCL and CLISTs for invoking Package Utility programs. The following is a summary of all ddnames reserved by the application. Note that some of the names may be shared with the FastLIST application.

1.	CIGOUT	Endevor exit output.
2.	CIGLOG	Utility program log dataset.
3.	CIGTRACE	Application trace ddname.
4.	CIGARCH	Archive log output.
5.	CIGRPT	Output ddname for all reports.
6.	CIGVTRAX	VSAM internals trace ddname.
7.	CIGPTRAX	PARSER internals trace ddname.
8.	CIGIN	Application input ddname.
9.	CIGINI	Initialization module override name.
10.	CIGINRDR	Internal reader ddname for AUTO-REMAKE.
11.	CIGJCLPK	JCL ddname for AUTO-REMAKE.

Cross-system VSAM Considerations

Cross-system VSAM Issues

Standard DASD Reserves are used for controlling access to the VSAM files. If you plan on using the Package Utilities or FastLIST across systems, we recommend that the VSAM access be managed by either by CA's MIM (Multi Image Manager) or IBM's GRS. To enable the Package Utilities to these products will involve an optional zap and definition to the reserve/enqueue tables.

Recommended MIM parameters

The MIM parameters recommended for the CIGQNAME are:

```
CIGQNAME
       GDIF=YES,
       SCOPE=SYSTEMS,
       EXEMPT=NO,
       ECMF=YES
```

LSERV Considerations

The Package Utilities Registry File is not managed by LSERV.

There is no problem running Endevor under LSERV and the Package Utilities not under LSERV.

INDEX

TABLE OF FIGURES