Chicago Interface Group, Inc.

FastLIST

Administrator Guide

Release 12.0

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

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

FastLIST release 12.0

Documentation version January 17, 2008

TABLE OF CONTENTS

FastLIST 12.0

Chapter 1: FastLIST Product Components

Introduction

FastLIST is a programmer's workbench for the CA-Endevor (referred to as Endevor throughout this document) environment. FastLIST consists of an ISPF end-user interface, a series of batch utilities, and a back-end process that executes as an Endevor exit. Central to FastLIST operation is the FastLIST database. The FastLIST database, a single-keyed KSDS VSAM file, provides the information to the various FastLIST components.

Figure 1.1 shows each FastLIST component interacting with the FastLIST database.

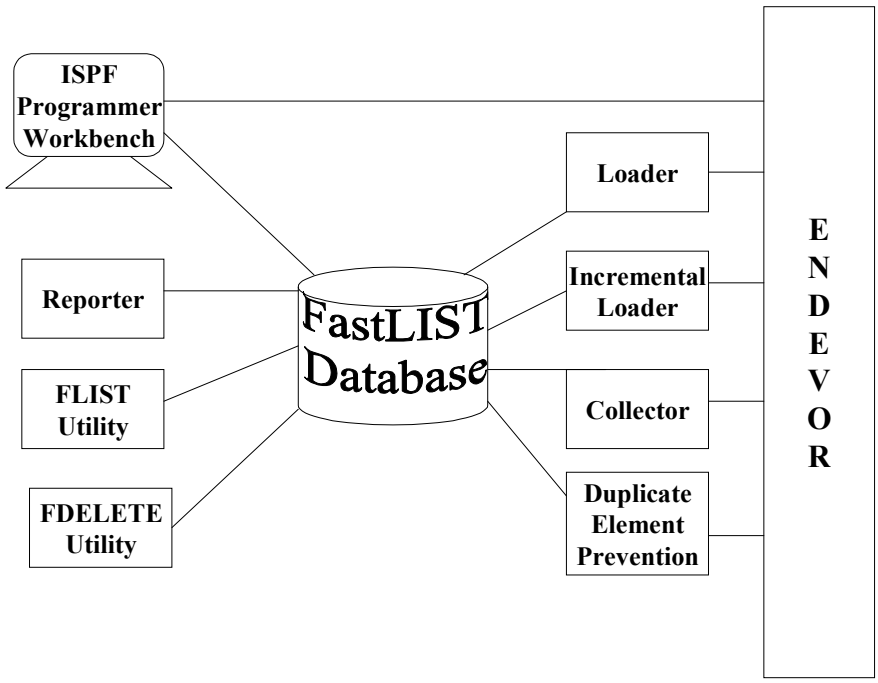


Figure 1.1
How the FastLIST Database is updated

FastLIST Components

❶	ISPF Programmer Workbench	Provides programmers with a simple to use, yet powerful tool to perform Endevor actions, edit Endevor elements, and perform impact analysis queries.
❷	Reporter	Reports on information contained in the FastLIST database.
❸	FLIST Utility	A High-speed replacement for the Endevor List Action.
❹	FDELETE	A batch utility which will delete information from the FastLIST database.
❹	Loader	A batch utility that performs an initial load of the FastLIST database.
❻	Incremental Loader	A batch utility that updates the FastLIST database with element information; similar to the Loader.
❼	Collector	Provides for real-time updating of the FastLIST database. The Collector executes as an Endevor exit.
❽	Duplicate Element Prevention	An Endevor exit, which prevents duplicate elements from being added into an incorrect Endevor location.

FastLIST Database Structure

The FastLIST database consists of relationships between various Endevor MCF and base/delta components. Information contained in the FastLIST database is organized to facilitate high-speed data access.

Figure 1.2 shows the FastLIST database structure.

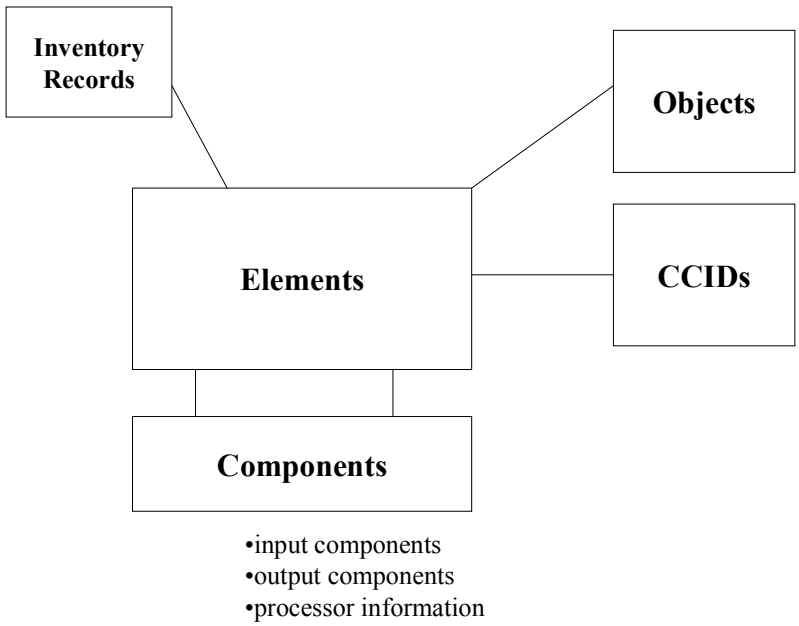


Figure 1.2
FastLIST Database structure

The FastLIST database structure shown in figure 1.2 provides various entry points when performing data access. For example, the following query can easily and quickly be satisfied: Show me all programs that use a copybook called CUSTCOPY. The ISPF and Batch Users Guides provide additional examples and instructions on querying the FastLIST database.

ISPF Programmer Workbench

The FastLIST Workbench provides programmers who need to access Endevor with a simple to use, yet powerful tool for performing Endevor actions, impact analysis, and editing. The FastLIST database is accessed by the FastLIST ISPF workbench to provide element lists based on queries by the end-user. Once a list of elements is displayed FastLIST can access Endevor source code and component data via a standard published Endevor interface.

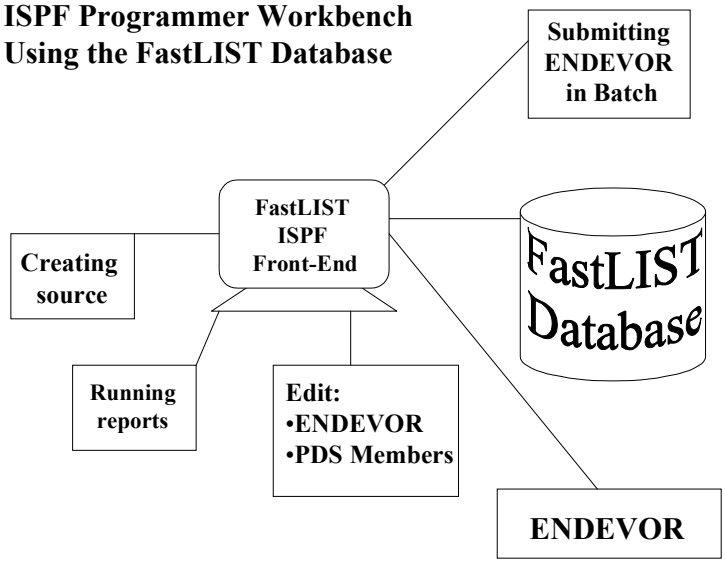


Figure 1.3
Using the FastLIST Database

Duplicate Element Prevention

The Duplicate Element Prevention Facility is invoked during Endevor exit 2 (Before Action exit point) processing. The FastLIST Duplicate Element Prevention Facility, invoked via the FastLIST exit driver called CIGFEXEC, utilizes Endevor exit blocks, the FastLIST database, and a duplicate element rules file to determine if a duplicate element condition exists The user-defined rules control whether to cancel an action, warn the user that a duplicate element exists, or ignore the condition.

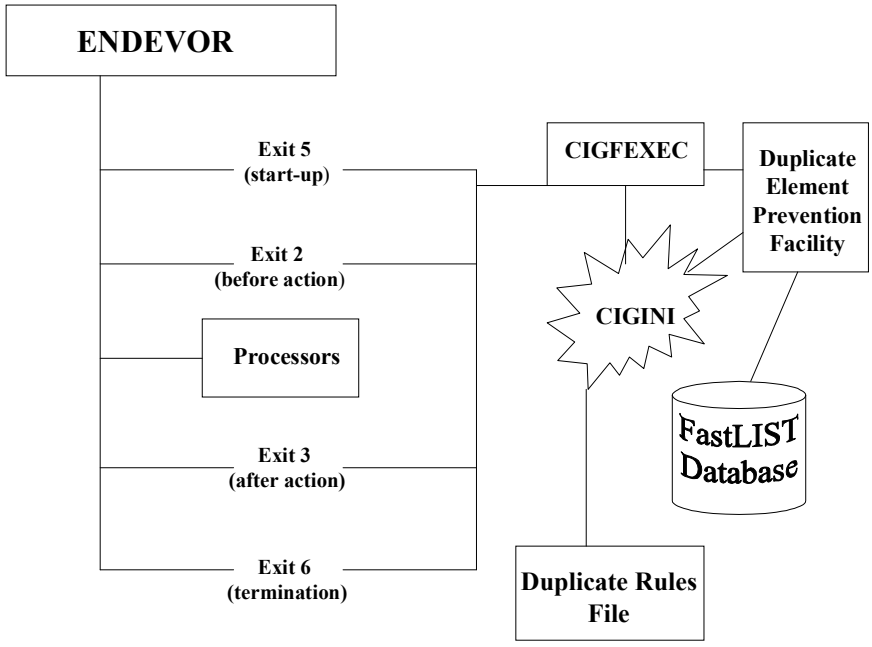


Figure 1.4
Duplicate Element Prevention Components

Duplicate Element Prevention Components

CIGFEXEC	This is a FastLIST Exit module defined to Endevor at exit points 2, 3, 5, and 6. This program will load the CIGINI module as well as the Duplicate Element Prevention facility. The CIGFEXEC as defined to the Endevor C1UEXITS table, must be located in the STEPLIB or LINKLIB.
CIGINI	This module contains FastLIST initialization parameters such as the product password; duplicate rules file location, CONLIB dataset name, and other initialization information. The CIGINI file must be compiled and link edited into the STEPLIB or LINKLIB, which is used during Endevor execution.
Duplicate Rules File	This file contains rules that describe a duplicate element condition. The Duplicate Rules File dataset name is defined as a parameter to the CIGINI file.
FastLIST Duplicate Element Prevention Facility	This facility is invoked via the CIGFEXEC program during Endevor exit 2 action processing. The Duplicate Element Prevention Facility is invoked by CIGFEXEC from the FastLIST product load library defined in the CIGINI file.
FastLIST Database	This file, also defined in the CIGINI file, contains element information used by the Duplicate Element Prevention Facility. The FastLIST database must be populated with element information in order for the duplicate element exit to operate.

FastLIST Loader

Initial Database Load

The FastLIST Loader Utility is used to initially load a FastLIST database. The utility is a multi-step batch job that populates the FastLIST database with data accessible via an Endevor SCL Print statement. A description of the key steps in the FastLIST Loader jobstream follows.

`Step 1: Process FLOAD Statements.` With FLOAD statements, database loading can be restricted by inventory location. Print element statements will be generated for all Endevor elements in the scope of the FLOAD statements.

`Step 2: Invoke Endevor.` Endevor will process the Print statements generated in Step 1 above.

`Step 3: Update the FastLIST database.` Output generated in Step 2 will be processed by the FastLIST loader. The FastLIST database is updated using the PRINT ELEMENT output as well as the initial FLOAD syntax request specified in Step 1.

`Step 4: Extract Endevor Data` Invoke the Endevor Batch Administration facility to extract System, Subsystem, Type, and Processor Group information.

`Step 4: Data Process Endevor Data` Run the CIGELOAD utility to load key Endevor control information.

Incremental Database Loader

The FastLIST incremental loader, like the FastLIST loader, is a batch utility used to update the FastLIST database. The incremental database loader, unlike the full database loader, will update the FastLIST database only if the element or component information in the Endevor database differs from the information in the FastLIST database. The steps and processing for incremental load processing is similar to the full load processing described above.

FastLIST Collector

The FastLIST database, once loaded with data using the FastLIST loader, can be kept up to date, real-time using the FastLIST Collector. The FastLIST Collector executes as an Endevor exit. Figure 1.5 illustrates invocation of the FastLIST Collector.

The FastLIST Collector is invoked during Endevor exit 2 and exit 3 processing. FastLIST Collector invocation occurs via the common FastLIST exit program called CIGFEXEC. The FastLIST Collector extracts Endevor information during Endevor exit processing and updates the FastLIST database.

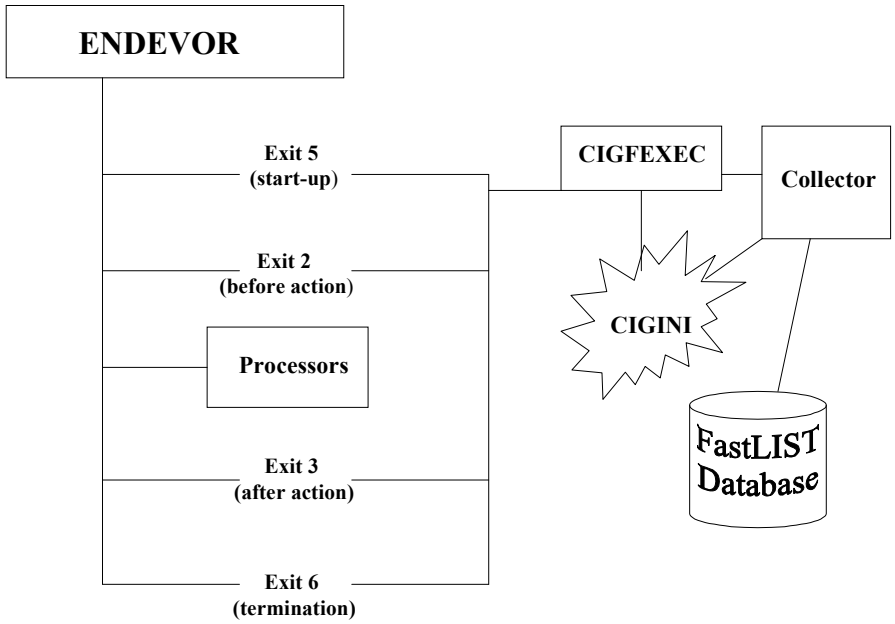


Figure 1.5
FastLIST Collector as a Back-end Process
To Endevor Action processing

FastLIST Collector Components

CIGFEXEC	This is a FastLIST Exit module defined to Endevor at exit points 2, 3, 5, and 6. This program will load the CIGINI module as well as the FastLIST Collector. The CIGFEXEC program, as defined to the Endevor C1UEXITS table, must be located in the STEPLIB or LINKLIB.
CIGINI	This table contains FastLIST initialization parameters such as the product password, duplicate rules file location, CONLIB dataset name, and other initialization information. The CIGINI file must be compiled and link edited into the STEPLIB or LINKLIB, which is used during Endevor execution.
Collector	This is the FastLIST facility that updates the FastLIST database real-time. The FastLIST Collector is loaded from the FastLIST product load library specified in the CIGINI file.

FastLIST and Endevor Exit Processing

All Endevor exits are implemented in the Endevor C1UEXITS exit table. Refer to Endevor documentation for a full description of exit implementation.

To simplify the FastLIST implementation, all FASTLIST exits invoke the same program called CIGFEXEC. This exit driver program will determine which FastLIST functions are to be invoked during Endevor exit processing.

Your installation may be using Endevor exit processing for other purposes. As such, you may already have defined exits to the Endevor C1UEXITS table. In order to ensure correct updating of the FastLIST database, carefully consider the order of invocation for exits. For example, if a local exit can cancel Endevor actions during exit 3 processing, invoke the local exit program before the CIG exit program to prevent incorrect information.

The following table illustrates the definition needed to identify FastLIST to the Endevor C1UEXITS table.

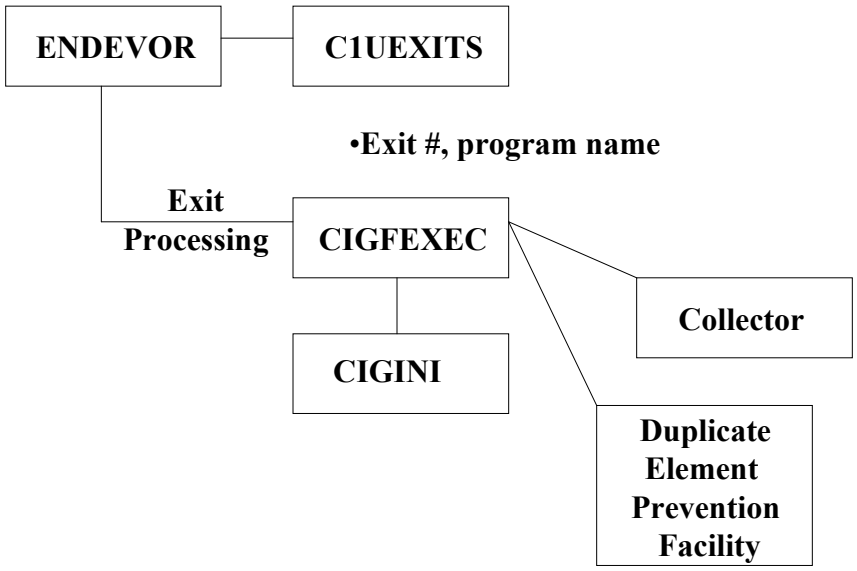


Figure 1.6
C1UEXITS Example

FastLIST Exit Processing Components

C1UEXITS	FastLIST exit processing requires the program CIGFEXEC to be defined to Endevor exits 2, 3, 5, and 6. (If you are using CIG Package Utilities, you must also define the CIGFEXIT program to Endevor exit 7.) Both the CIGINI modules and the CIGFEXEC modules must be copied into the STEPLIB used by Endevor or copied into a link listed library.
CIGINI	FastLIST exit programs are loaded by the program CIGFEXEC based on the product load library parameter specified in the CIGINI module. The CIGINI module is a load module that contains FastLIST initialization parameters. Both the CIGINI module and CIGFEXEC module must be copied into the STEPLIB used by Endevor or copied into a link listed library.
FastLIST Collector And Duplicate Element Prevention Facility.	These product components are loaded from the product load library specified in CIGINI module.

The CIGINI module is similar to the Endevor C1DEFLTS module. The CIGINI module contains product initialization parameters such as the FastLIST product password and FastLIST database name. The CIGINI module must be copied into the STEPLIB or LINKLIB used by Endevor.

FastLIST 12.0

Chapter 2: Implementation

Implementation

Overview

FastLIST is a set of tools designed to work with Endevor data. It has been designed as a workbench product that provides maximum flexibility in terms of usage and implementation. This flexibility is required because Endevor implementations vary from customer to customer, or even within a single company.

FastLIST provides for multiple initialization modules to allow alternate database and Collector options within a single configuration. The beauty of FastLIST's flexible structure is that you need only one CIGINI module to get FastLIST up and running, but as you enable additional Endevor users and their Endevor inventory to FastLIST, you can provide a variety of options and alternate databases. CIG recommends that you start with one database and, as you get familiar with FastLIST, create alternate databases and override initialization modules. Also, remember that at any time, you can change your implementation of FastLIST by running FLOAD and reloading your FastLIST database.

To implement and enable FastLIST, perform the following steps:

STEP #	ACTION
❶	Determine inventory and strategy.
❷	Review options and build the CIGINI module.
❸	Allocate the FastLIST database.
❹	Enable the Collector.
❺	Load Endevor with pilot inventory.
❻	Run the cross-reference reports.
❼	Set up database maintenance job.

Figure 2.1
Implementation Step-by-Step

Implementation Examples

In Examples 1-4, in the Endevor implementation, three mapped environments comprise a standard application life cycle. The environments are named "Test," "QA," and "Prod" (Production). In addition, in these four examples, all users access Endevor via the same C1DEFLTS table.

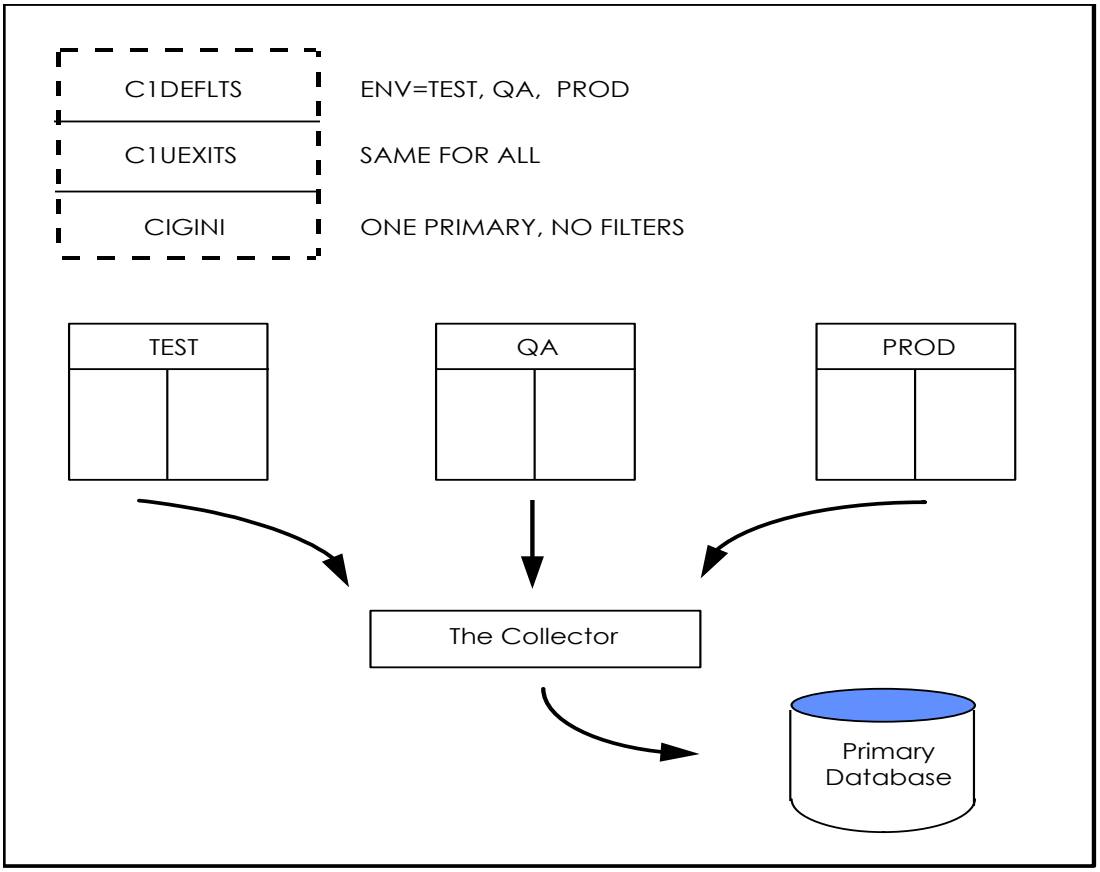


Figure 2.2
Implementation Example 1

Example 1 (figure 2.2) - The FastLIST implementation uses only a single, primary CIGINI file and its corresponding primary database. The inventory from all three environments is loaded into the FastLIST database using FLOAD. Because one database holds the entire inventory, the FILTER option in the CIGINI file is not used, making all elements eligible for collection.

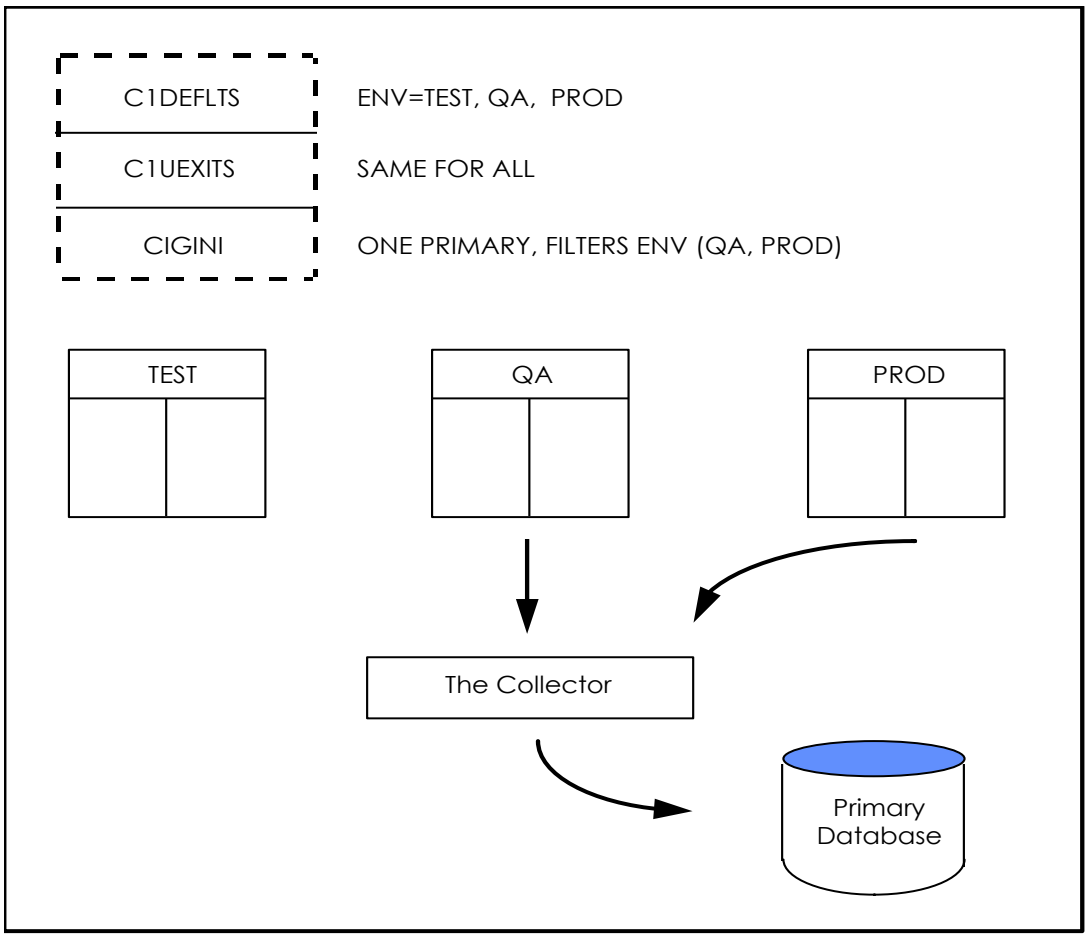


Figure 2.3
Implementation Example 2

In Example 2 (figure 2.3), the FastLIST implementation is slightly different. It still uses a single primary CIGINI file and its corresponding primary database, but the TEST environment is not included in the FastLIST database during the FLOAD. To prevent unwanted elements in the TEST environment from entering the database via the Collector, the CIGINI module contains the optional FILTER ENV (QA,PROD) syntax.

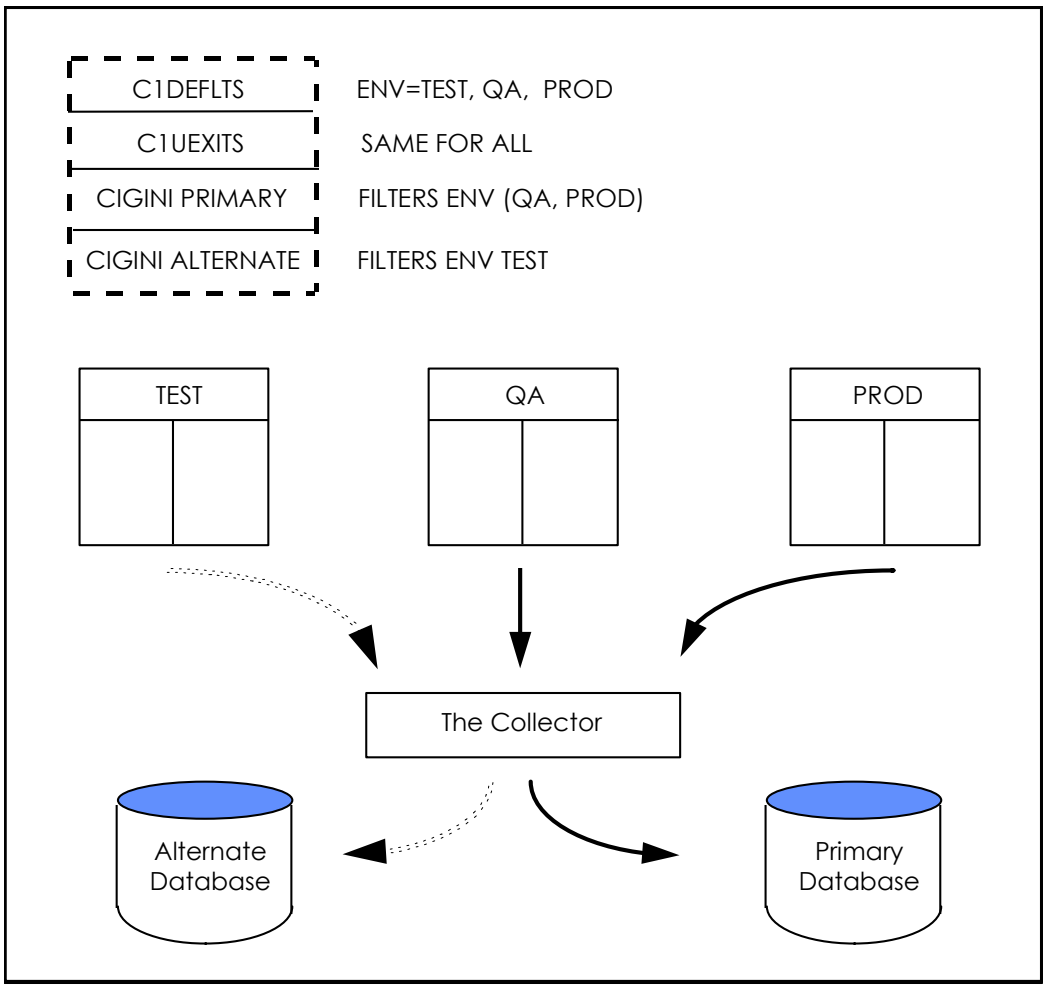


Figure 2.4
Implementation Example 3

As in example 2, a primary CIGINI file and its corresponding primary database are established in example 3 (figure 2.4), and the CIGINI module contains the optional FILTER ENV (QA,PROD) syntax. Additionally, this implementation has created a separate FastLIST database for the TEST alternate CIGINI module. This file contains the syntax FILTER ENV TEST, as well as syntax which points to the alternate VSAM database. All users working in the TEST environment must code a CIGINI DD statement pointing to the alternate module.

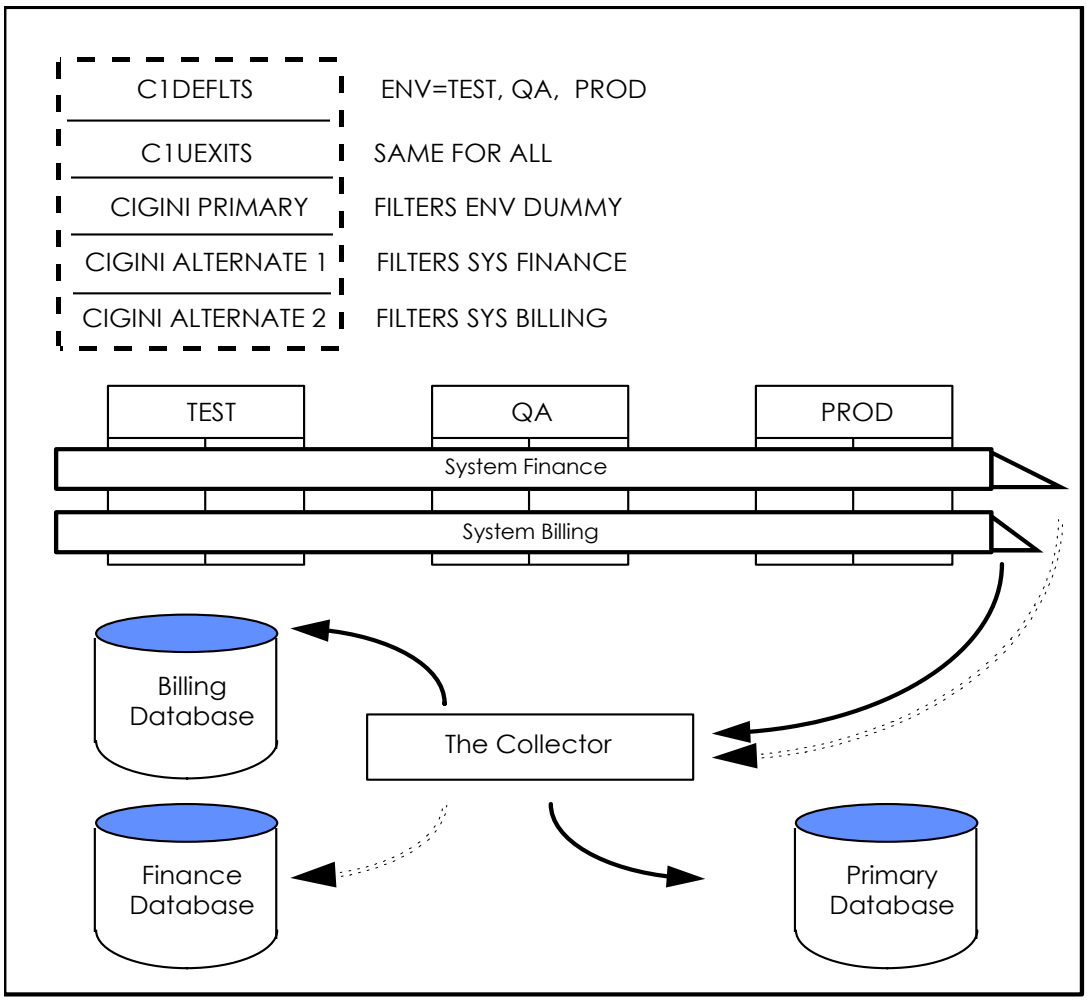


Figure 2.5
Implementation Example 4

The implementation in example 4 (figure 2.5) is structured differently than in the previous examples. Here, the Endevor implementation has two primary systems, FINANCE and BILLING. Each system has its own override module and database. For password check purposes, the primary CIGINI file and database were installed and allocated, but the primary CIGINI file includes the syntax FILTER ENV DUMMY, resulting in an empty primary database. To ensure that the Collector updates only the proper database, the override modules contain the syntax FILTERS SYSTEM FINANCE and FILTERS SYSTEM BILLING, respectively.

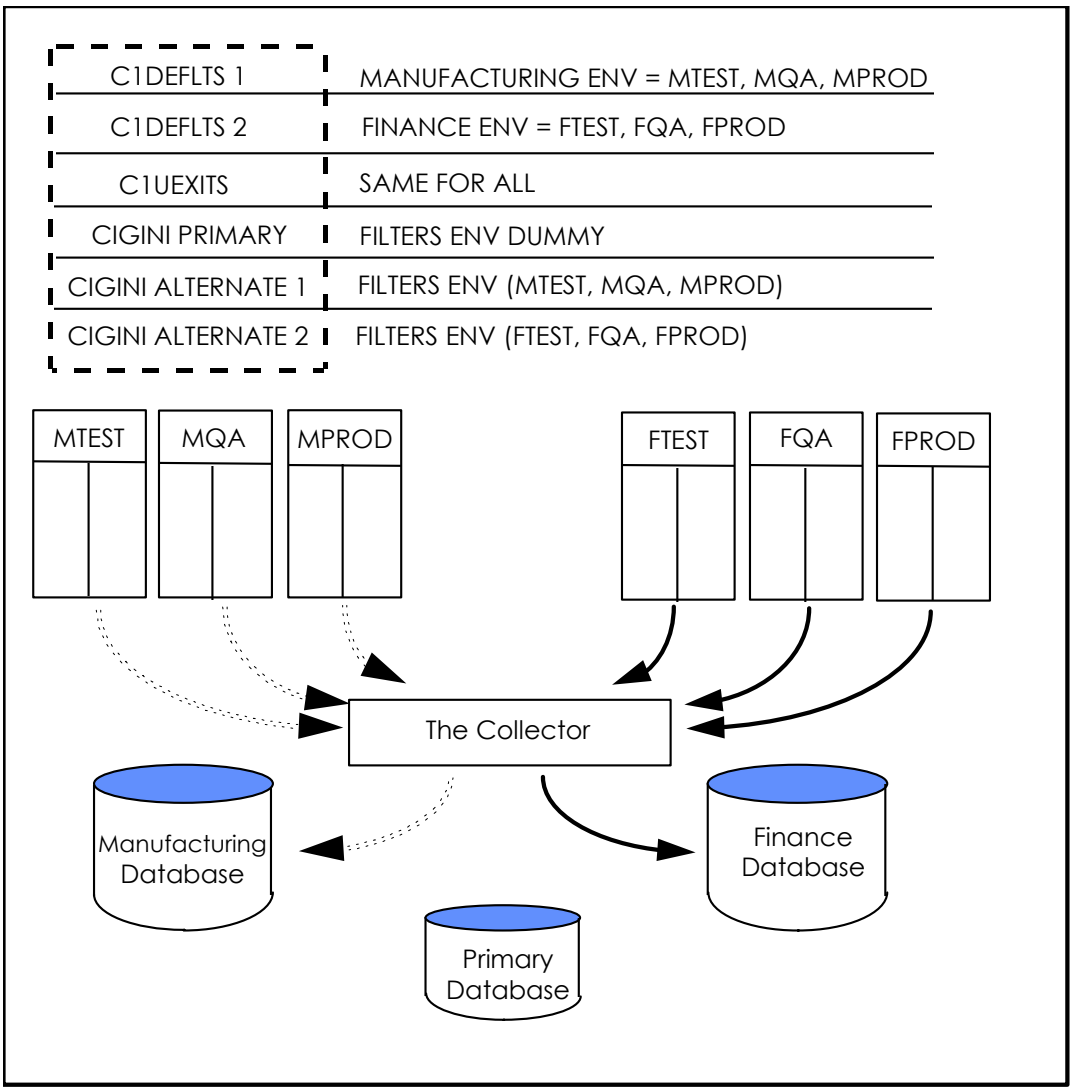


Figure 2.6
Implementation Example 5

Example 5 (figure 2.6) - In this implementation, we see for the first time multiple C1DEFLTS tables, creating two separate software life cycles, one for manufacturing and one for finance.

Again, because a primary CIGINI is required, the primary CIGINI file and database were installed and allocated, but the syntax FILTER ENV DUMMY is included in the primary CIGINI file, resulting in an empty primary database. Two alternate CIGINI modules were set up, one containing the syntax FILTERS ENV (MTEST, MQA, MPROD), and the other containing FILTERS ENV (FTEST, FQA, FPROD). The alternate CIGINI modules also point to their respective VSAM database. At this point, all of the setup work has been done, and you are ready to load the pilot inventory into the FastLIST database.

To ensure a clean load and collection, make sure that the inventory loaded by FLOAD is the same inventory coded in the CIGINI FILTERS. FastLIST will not allow you to load any inventory into your database that does not satisfy the values coded in your CIGINI FILTERS.

The Endevor inventory should be loaded at a time when the system is dormant. This step primes your databases with the current configuration and historical CCID data. Once the system is loaded, the Collector keeps it synchronized with Endevor by updating the database per action executed.

The steps of Loading the inventory and enabling the Collector should be performed together. The Collector immediately begins to update the database once enabled. The Collector should start working against a database that has been primed with a "level set" of inventory data by FLOAD.

Incremental Updates

Users also have the option of keeping the database in sync through the use of the incremental loader. The incremental loader will perform an FLOAD only on those elements that have changed. The syntax and JCL are the same as for the regular (full) FLOAD, with the addition of an INCREMENTAL keyword in the syntax block. This method is an alternate to enabling the Collector.

Sample FLOAD Syntax

Figure 2.7 shows *flhq1.flhq2.*SAMPLIB(FLOADIVP) which contains a sample of the syntax to load an entire ENVIRONMENT into FastLIST. Modify this syntax to specify the inventory you wish to implement. You will use the JCL member called CIGJCL05 to run the FLOADIVP syntax. The member CIGJCL05 is contained in the FastLIST product JCL library.

```
FLOAD ELEMENTS *
  FROM ENV TEST
    SYS  *
    SUB  *
    TYPE  *
    STAGE NUM  *
              .
```

Figure 2.7
FLOAD Sample Syntax

Defining FastLIST Run-Time Options

CIGINI Options

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

1. The name of the FastLIST VSAM File.
2. The name of the FastLIST LOADLIB.
3. FastLIST Options.
4. FastLIST FILTERS.
5. Product password.
6. VIO and Work units.

Your installation must build a primary CIGINI module and place the module in a STEPLIB or LINKLISTLIB library. Like the Endevor C1DEFLTS module, the CIGINI module will be loaded at initialization and used while FastLIST is active. The CIGINI module is always used for the password check, and contains the default initialization parameters for FastLIST. Even if you determine that you want more than one FastLIST database, 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

You should access the member called CIGINI located in the FastLIST product SAMPLIB and change the variables to reflect your desired options for your primary CIGINI module.

Sample CIGINI Source

```
************************************************************
*  SAMPLE CIGINI FILE
************************************************************
DEFINE COMMON SECTION
  PRODUCT LOADLIB     = 'FLHQ1.FLHQ2.LOADLIB'
  WORK UNIT           = WORK
  VIO  UNIT           = WORK
  DO NOT ALLOW ALTERNATE CIGINI FILE
  Endevor CONLIB DSNAME = 'QUAL1.QUAL2.CONLIB'

************************************************************
*  DELETE THIS SECTION IF NOT INSTALLING FASTLIST.
************************************************************
DEFINE FASTLIST SECTION
  PASSWORD   = 'PASSWORD'
  VSAM DSNAME = 'FLHQ1.FLHQ2.DEMODB'
```

Figure 2.8
Sample CIGINI Source

Turning on the Collector

Modify Job CIGJCL06

The FastLIST Collector, via exit processing, keeps the FastLIST database synchronized with Endevor. To enable the Collector, define the CIG Endevor exit program CIGFEXEC to your Endevor C1UEXITS table and reassemble the C1UEXITS table. If your current Endevor implementation includes user exits, review your existing C1UEXITS table to determine where the FastLIST exit program should be defined. It is recommended that the FastLIST exit module be placed first in the user exit list of C1UEXITS unless an existing exit program can cancel an Endevor action.

If your current Endevor implementation does not include any user exits, build a C1UEXITS input table as per your Endevor documentation. Note that the C1UEXITS table must reside in the Endevor authorized library. For more information on how to implement the Endevor Exits Table, see the Endevor Exits manual.

Allocating the Database

Introduction

By this point you should have identified the Endevor inventory to be loaded into the FastLIST database, and coded and built the CIGINI file to reflect the options and inventory chosen. To estimate the size of a production FastLIST database, complete the following worksheet. After calculating the space estimate, multiply it by an estimated growth factor to account for new elements added to Endevor. Finally, the standard freespace parameter for the FastLIST database is 30 percent. To account for this space (reserved by VSAM during database reorganization) multiply the size estimate by 1.4.

FastLIST Database Size Worksheet

Step I: Estimate number of Endevor elements and components:

Number of Elements to be loaded into the FastLIST database _____ELM

Average number of components per element(use 20 as an estimate) _____COMP

Average number of CCIDs per element (use 10 as an estimate) _____CCID

Step II: Using estimates from Step I, estimate database space requirements:

COMP ( ) x 600 bytes _____

CCID ( ) x 500 bytes +_____

Subtotal: =_____

Times # of elements: ELM ( )

Component and CCID subtotals =_____

ELM ( ) x 600 bytes +_____

Total database size in bytes =_____

TRACK size ÷ 56,000

Number of Tracks =_____

Cylinders per track ÷ 15

Number of Cylinders =_____

Multiple Databases and CIGINI Modules

Multiple Database Issues

The power of FastLIST comes from its ability to take advantage of the data produced by Endevor. For your installation's implementation of FastLIST, you must decide how many databases and what CIGINI options to use for each database.

FastLIST provides three mechanisms to implement multiple databases. The first is to have multiple primary CIGINI files. This is only possible if each group of users has entirely separate STEPLIB concatenations and will always use these concatenations for all batch and online executions. CIG does not recommend this approach.

The second approach is to define a single primary CIGINI module and one or more "Override CIGINI" modules. This approach has been described above and is detailed further in the following section. This approach is approved, but the following approach is recommended where possible.

The third approach is to use the "FOR" clause in conjunction with CIGINI filter statements. This implementation is documented in this section and is the recommended approach.

Separate FastLIST databases must have inventories that are completely distinct. If two separate FastLIST databases contain the same element, then the two databases will not remain synchronized with the information in Endevor. This is because the FastLIST Collector will only update one database when actions are performed on the common element.

The following are some issues to consider when determining whether to use a single database or multiple database strategy for total implementation:

- Perform an analysis of your existing inventory. Is the inventory segmented and isolated by applications, environments, or systems?

  If you want to provide separate databases with different options, then segmented inventory makes separate databases easier to implement. If your implementation strategy is to have all FastLIST users access the same database, then inventory segmentation is not an issue.

- Do all Endevor users use MCF and DELTA CCIDs and ACM?

For each individual group of users, consider whether they use configuration management information (component data), MCF CCIDS, DELTA CCIDs, or a combination of these.

- Aside from Endevor inventory issues, how separate and independent are groups in terms of physical access to Endevor and logon procedures?

  If the groups each have their own STEPLIB, then a separate primary CIGINI module would work for this type of software configuration.

  If, based on the above considerations, you determine that you need more than one FastLIST database, and then you will need to do one of the following. Create override modules to point to these alternate databases, create multiple CIGINI primary modules for separate STEPLIB conditions, or define multiple databases to the CIGINI module via the FOR clause.

  Because your implementation strategy will need to grow as your Endevor inventory grows and changes, FastLIST implementation is simple, flexible, and expandable. Remember that you do not need to know your ultimate implementation strategy prior to getting one group up and running. You may always add additional elements to an existing database, or add additional databases to an existing implementation.

Alternate CIGINI

If you determined that you want more than one FastLIST 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, FLIST, FLOAD, and the ISPF front end.

2. There must always be at least a primary CIGINI module in a STEPLIB or LINKLIB. 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 FastLIST database containing different Endevor inventory or having different options, you must create an alternate initialization module containing the desired parameters. Remember that inventory in the separate databases must be completely distinct. If there is any inventory overlap between databases, FastLIST will be unable to keep the databases synchronized with Endevor.

4. The alternate initialization modules should be named appropriately for their usage. There are no naming standards enforced by FastLIST, 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 alternate initialization module, the application will be terminated.

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

Multiple Databases via the FOR clause

You can also define multiple FastLIST databases via the FOR clause within the CIGINI file. The FOR clause allow you to direct database updates to different FastLIST databases based on inventory specifications. A description of the FOR clause is located in the Utilities chapter of this manual.

Multiple Databases and Duplicate Element Prevention Facility

The Duplicate Element Prevention Facility operates against only one database per action. As such, careful attention must be given when utilizing alternate CIGINI files or multiple databases via the CIGINI FOR clause.

Database Integrity

Production considerations for the CIG databases

The FastLIST database is a production VSAM file that needs to be allocated, sized, and maintained as such. Prior to going into production with FastLIST, the following must be done:

Set up the backup and restore job. (CIGDBJ04) Make sure it is a scheduled production job. Test the job. For active files, it is recommended that the file be backed up and reorganized nightly.

Test security on the file. Have an end-user log on to the product and test access to the file. Quite often, the person installing or implementing the products has a higher security profile than the average user. FastLIST database security is discussed later in this chapter.

Review the MIM or GRS parameters and usage. If your VSAM file is on a pack that is shared cross-system, then you will need to define CIG files to GRS or MIM. See the following discussion on cross-system VSAM considerations.

Review the allocation of the production database. Make sure the FastLIST database has room to grow. We recommend a large primary allocation to guarantee space availability.

Reorganize the database after the initial database load. After the FLOAD process has been completed, perform an IDCAMS reorganization using the CIGDBJ04 JCL modified previously. Review the file usage and number of extents. Tune the size allocations to prevent secondary allocations. This will limit the number of extents used by the file and ensure maximum performance for your FastLIST implementation.

Set up a CIGJCL07 for recovery purposes. CIGJCL07 should be tailored and ready to perform incremental loads. Make sure the various temporary files are sized for maximum possible values and that the region parameter and job class are appropriate for long running batch jobs.

Set up FastLIST diagnostic aids. Set up the PRINTINI CLIST for better diagnostic abilities.

Database Backup, Reorganization, and Recovery

The JCL member CIGDBJ04 is used to reorganize, backup, and restore the FastLIST database. A regularly scheduled job to reorganize the VSAM file is required to ensure optimal performance.

Regularly scheduled database reorganization jobs are of additional importance with FastLIST due to logically deleted records written to the FastLIST database during action processing. As elements are promoted from stage to stage through the Endevor life cycle are the FastLIST database records are tagged as logically deleted. By utilizing logically deleted records, the database update activity to the data index portion of the VSAM file is greatly reduced. However, logically deleted records in the FastLIST database need to be eliminated on a regular basis or else the FastLIST database will grow in size.

Logically deleted records will be reused if an element is added back to the original Endevor inventory location, but this does not justify leaving the excess records in the database.

The CIGDBJ04 utility will both eliminate FastLIST logically deleted database records, as well as eliminate any CI and CA splits which may have occurred during VSAM processing.

Cross-System VSAM Considerations

Standard ENQ/DEQ macros are used by FastLIST to ensure file integrity. When database updating occurs, FastLIST will issue an ENQ macro with the option SYSTEMS. To ensure the ENQ is propagated across CPUs and LPARs, we recommend that the FastLIST databases be managed by either GRS from IBM or CA-MIM (Multi-Image Manager) from Computer Associates.

Example of QNAME definition:

GRS

```
RNLDEF   RNL (INCL)      TYPE   (GENERIC)
QNAME    (CIGQNAME)

RNLDEF   RNL (CON)       TYPE (GENERIC)
QNAME    (CIGQNAME)
```

MIM

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

QNAME/RNAME

The QNAME used by FastLIST is called CIGQNAME. The RNAME is the name of the FastLIST database VSAM cluster.

Security Considerations

The following summarizes FastLIST database access authority:

FastLIST Collector	Update authority. Access controlled based on the value in the RACFUID= value in the C1DEFLTS table. If Endevor Alternate ID processing is not used the invoking user id must have update authority.
FastLIST Loader	Update authority required by job submitter.
FastLIST ISPF front-end	Read authority.
FLIST utility	Read authority.
FastLIST Reporter	Read authority.
FDELETE utility	Update authority by job submitter.
CIGVSM2L utility	Update authority by job submitter.

FastLIST Collector

CA-Endevor 3.7.2 or later

When running in a CA-Endevor release 3.7.2 environment, FastLIST Collector security requirements are controlled by the same rules that control access to key Endevor files such as the Endevor's Master Control File (MCF) and the Endevor base and delta files.

Under Endevor release 3.7.2 and later, access to the FastLIST database by the FastLIST Collector is controlled based on the RACFUID= value specified in the Endevor C1DEFLTS table.

If no RACFUID= value is specified in the Endevor C1DEFLTS table then file access by the FastLIST Collector occurs based on the TSO user id or batch job card information.

CA-Endevor 3.7.1 or prior

Access to the FastLIST database by the FastLIST Collector is performed under the authority of the TSO user id or batch job card information. As such, all users who were accessing CA-Endevor are required to have update and read access authority to the FastLIST database.

FastLIST Loader, Incremental Loader, and FDELETE Utility

Batch jobs that invoke the FastLIST Loader and Incremental Loader must have update access authority to the FastLIST database.

FastLIST ISPF front-end, FLIST Utility, and FastLIST Reports

Users accessing the FastLIST database via the above-listed FastLIST components are required to have read access authority to the FastLIST database.

CIGVSM2L Utility to expand databases to two levels

Users submitting the index level expansion utility are required to have update authority to the FastLIST database.

Authorization Requirements

With the exception of the FastLIST Collector, there is no requirement to run FastLIST from an authorized library.

The FastLIST Collector requires that two modules be placed in an authorized library: CIGFEXEC and CIGINI. Both modules run in a non-authorized mode; both modules must be located in the Endevor STEPLIB or in a link-listed library. All other FastLIST load modules are located in the dataset as specified in the CIGINI module.

The figure below illustrates the relationship between the modules CIGFEXEC, CIGINI, and the other FastLIST product modules.

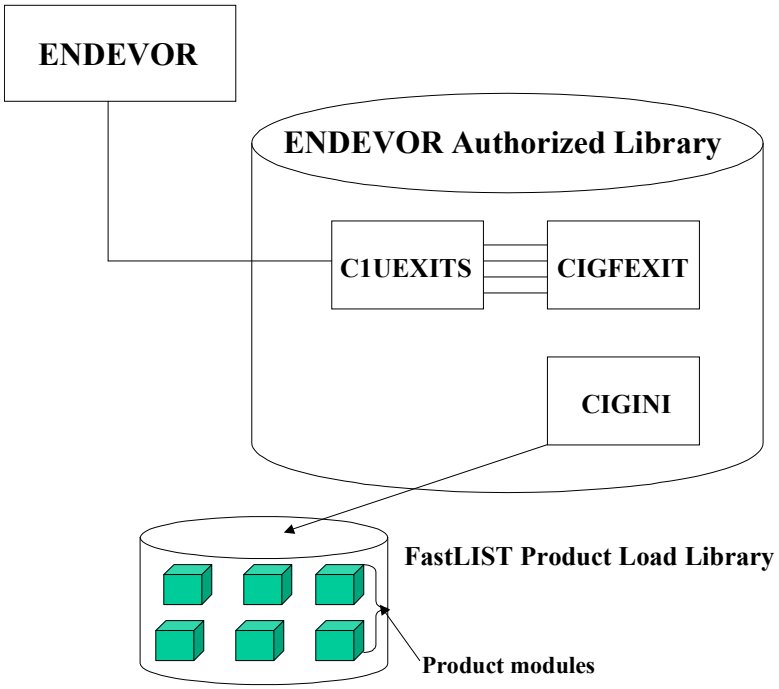


Figure 2.9
Relationship Between Modules

FastLIST Utilities in Endevor Processors

The FLIST Utility was designed as a high-speed replacement to the Endevor LIST action. When defining the FLIST utility or any other FastLIST utility inside an Endevor processor it is important that the utility program name be specified on the EXEC PGM= JCL statement. Do not execute the program CIGFEXEC.

The following illustrates use of the FLIST utility in an Endevor processor:

```
//FLIST     EXEC PGM=CIGFLIST
//STEPLIB    DD   DSN=f1hq1.f1hq2.LOADLIB,DISP=SHR
//CIGLOG     DD   SYSOUT=*
//CIGIN      DD    *
           FLIST ELEMENT  *
                 WHERE INPUT COMPONENT = &C1ELEMENT.
```

Figure 2.10
FLIST Utility in a Processor

Other Considerations

Running Multiple CIG Products

Chicago Interface Group products are designed to work independently of each other as well as to coexist in an Endevor environment. At the same time, CIG products share core technology modules and are co-dependent. CIG recommends that a single set of product libraries be created to implement all enabled CIG products.

Endevor Compatibility

All CIG products operate with Endevor release 3.7 or later.

LSERV Considerations

The FastLIST database is not managed by LSERV. There is, however, no problem in defining Endevor's Master Control File (MCF) to LSERV and not defining the FastLIST database to LSERV.

FastLIST 12.0

Chapter 3: FLOAD - The Database Loader

Introduction to FLOAD

Overview

The FLOAD utility extracts information from Endevor, formats it into indexed structures, and loads it into a FastLIST database. FLOAD will load as much or as little of your inventory as you desire. You can load entire environments, as when initially loading your database, or as little as a single element when performing an incremental update of your existing database.

You have the option of performing an incremental load, which will only load those elements that have changed or a full load, which is an unconditional. You also have a facility called FDELETE, which deletes segments of inventory from the database.

Full Load

Whether you choose to perform a full load or an incremental load, the amount of data loaded into the database is up to you. It is best to load inventory in manageable increments, e.g., by Environment or by System and/or Type within an Environment. After each load, FLOAD creates an execution summary report. There is an optional detailed line report available by allocating the CIGRPT dataset. This detailed report contains a line for each record written to the FastLIST database.

When you initially load your database, you will perform a mass load into an empty database. Running this mass load provides a "level-set" of current configuration and CCID information, which is then kept synchronized with Endevor by the Collector. You can later perform incremental loads if, for instance, you recover the FastLIST database and now need to update the records with Endevor changes since the backup was taken.

DO NOT run a full load on an existing FastLIST database. A full load will first delete all current information in the load request and then perform a load.

Incremental Load

If you do not want to use the Collector, you have the option of periodically running an incremental load, which will selectively update the elements that have changed since the last load. The JCL used to perform an incremental load is the same that you use to perform a full load. The key word “incremental” on the FLOAD statement tells the FLOAD Utility to perform an incremental load, as shown in figure 3.4.

Loading the Database Step-by-Step

Follow these steps to load or delete elements from the FastLIST database.

❶	Review initialization parameters and determine inventory to load or delete.
❷	Modify FLOAD JCL.
❸	Code FLOAD syntax.
❹	Submit JCL.
❺	Review the output reports and return codes from the job output.

Figure 3.1
FLOAD Step-by-Step

Setting up for FLOAD

Associated Initialization Parameters

These CIGINI initialization parameters will affect the FLOAD utility. All of these parameters are located in the CIGINI initialization file. For more information about setting these values, see chapter 6.

[DO NOT] COLLECT CCIDS

Specifying **[DO NOT] COLLECT CCIDS** in your CIGINI file directs FLOAD to exclude all CCID information from the FastLIST database.

COLLECT MASTER CCIDS ONLY

Specifying **COLLECT MASTER CCIDS ONLY** in your CIGINI file directs FLOAD to exclude delta- level CCID information from the FastLIST database.

[DO NOT] COLLECT COMPONENTS

Specifying **[DO NOT] COLLECT COMPONENTS** in your CIGINI file directs FLOAD to exclude all component information from the FastLIST database.

FILTER

The use of inventory **FILTERS** will control the inventory loaded into the database as well as the inventory monitored by the Collector.

Associated ddnames, Descriptions, and Attributes

The following are the ddnames used by FLOAD. Ensure that all required datasets are allocated with the proper attributes. Note that the FLOAD JCL on the following page contains many required ddnames that belong to Endevor. For these ddnames, use standard Endevor attributes. Note that the JCL provided to execute the FLOAD utility makes extensive use of temporary datasets for passing information between steps. Do not modify the names, attributes, or ddnames of these temporary datasets.

ddname	Description	Attributes
CIGIN	***REQUIRED* for Steps 1, 3, 5** The input dataset containing FLOAD syntax. If the dataset is not found or cannot be opened, FLOAD fails in initialization.	LRECL=80 RECFM=FB DSORG= PS or PO
CIGLOG	***REQUIRED* for Steps 1, 3, 5** FLOAD messages are written to this dataset. If the dataset is not found, or cannot be opened, FLOAD fails in initialization.	DSORG=PS LRECL=132 BLOCKSIZE=13200 RECFM=FBA or SYSOUT = *
CIGRPT	***OPTIONAL*** Use this optional dataset in STEP5 for a detailed line report on FLOAD.	LRECL=132 BLOCKSIZE=13200 DSORG=PS RECFM=FBA Or SYSOUT=*
CIGINI	***OPTIONAL*** Required in all steps, if initializing from an override CIGINI module (See chapter 3, *Implementation*). Dataset must reference a member name.	DSORG=PO LRECL=0 RECFM=U BLOCKSIZE=6233
CIGTRACE	***OPTIONAL*** Allocating this CIGTRACE writes additional informational and debugging messages to the CIGLOG DD.	DUMMY

Figure 3.2
Associated ddnames, Descriptions, and Attributes

JCL to Run the FLOAD Utility

The table below shows the sample FLOAD JCL streams in the CIG JCLLIB offloaded from the installation tape. Use this table to determine which JCL is appropriate for your site and products installed. The abbreviation GH will be used to indicate the Greenhouse product.

Member name	Load Type		FastLIST DB for		Loads 'EE'		Comments
	Full	Incr.	Endevor	GH	Yes	No	
CIGJCL07	X	X	X	X	X		Full JCL - all steps
CIGJCLF7	X		X		X		No Greenhouse steps
CIGJCLE7			X		X		Only loads Endevor Env. records

JCL Members to Run FLOAD

JCLLIB members CIGJCLF7 and CIGJCLE7 are shortened versions of CIGJCL07. These jobs reflect only the specific steps needed for the documented functions.

Determine which JCL is appropriate for your site. Tailor and submit the appropriate JCL to run FLOAD. For Endevor ddnames, use standard Endevor attributes. Running FLOAD in a processor is not recommended.

The standard CIGJCL07 job (shown in figure 3.3) includes two steps (and associated IF – ENDIF statements) that are required for maintaining element tags associated with the CIG Greenhouse product. If the Greenhouse product is not in use at your site these steps can be removed, however they will not affect processing if they remain in place.

The standard CIGJCL07 job also includes two steps (and associated IF - ENDIF statements) that refresh the Endevor inventory information records in the FastLIST database. These steps can be removed or suppressed. JCLLIB member CIGDBJ15 contains these steps in a stand-alone job that can be run independently of the FLOAD process.

```
//**(JOBCARD)
//**
//*    -------------------------------------------------------------------*
//*  NAME:    CIGJCL07                                                     *
//*   PURPOSE: PERFORM LOAD OR INCREMENTAL LOAD OF FASTLIST DATABASE.      *
//*    -------------------------------------------------------------------*
//*  TO USE THIS JCL, YOU MUST:                                           *
//*           1) INSERT A VALID JOB CARD WITH VALID CLASS AND REGION=0M   *
//*           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 YOUR ENDEVOR             *
//*              CONLIB DATASET                                           *
//*           4) CHANGE FLHQ1, FLHQ2, QUAL1 AND QUAL2                     *
//*              AS PER YOUR INSTALLATION WORKSHEET                       *
//*           5) CHANGE THE UNIT=TDISK TO THE APPROPRIATE UNIT            *
//*              NAME.                                                    *
//*          6) REVIEW SYNTAX IN STEP SYNTAX.  MODIFY TO REFLECT THE   *
//*             TYPE OF LOAD (ADD INCREMENTAL STMT BEFORE '.') AND THE *
//*             INVENTORY TO BE LOADED                                  *
//*           7) MAKE SURE THAT IF YOU ARE USING AN OPTIONAL CIGINI      *
//*              DD THAT YOU INCLUDE THE DD STATEMENT IN EACH OF THE     *
//*              5 STEPS.  SEE THE MANUAL FOR OTHER REQUIRED DDNAMES     *
//*           8) FOR NON-GREENHOUSE USERS, DELETE THE CIGSNAP DDNAMES,   *
//*              FROM STEPS STEP1 AND STEP3.  REMOVE JOB STEPS STEP5A    *
//*              AND STEP6.  DO NOT REMOVE ANY IF OR ENDIF STATEMENTS.   *
//*           9) MODIFY THE BUILD SCL REQUEST IN STEP7 TO REFLECT THE    *
//*              SCOPE OF YOUR FASTLIST INSTALLATION.  OPTIONALLY,        *
//*              THESE STEPS CAN BE REMOVED FROM THE LOAD JOB AND RUN     *
//*              SEPARATELY.  JCL TO EXECUTE THESE STEPS IS IN MEMBER     *
//*             CIGDBJ15 IN THE CIG PRODUCTS JCLLIB.                      *
//*          10) CHANGE THE FASTLIST DATABASE NAME IN STEP 8 TO THE       *
//*              DATABASE TO BE UPDATED.                                  *
//*    -------------------------------------------------------------------*
//* Z180214A IMPROVE PERFORMANCE OF INCREMENTAL LOADER
//*    -------------------------------------------------------------------*
//*  SYNTAX: ALLOCATE AND POPULATE THE SYNTAX DD HERE.                     *
//*    -------------------------------------------------------------------*
//SYNTAX   EXEC PGM=LISTFILE
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//SYSIN    DD  *
FLOAD ELEMENTS *
 FROM ENV YOUR-ENVIRONMENT-NAME
      SYSTEM     *
      SUBSYSTEM  *
      TYPE  *
      STAGE NUMBER  *
.
//SYSOUT   DD  DSN=&&SYNTAX,
//             SPACE=(TRK,(1,1)),
//             DCB=(BLKSIZE=31200,LRECL=80,RECFM=FB,DSORG=PS),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//* ------------------------------------------------------------------*
//*   STEP1:   READ FLOAD STATEMENTS AND WRITE ENDEVOR PRINT STATEMENTS *
//*           RC=0  STANDARD FULL LOAD                                  *
//*           RC=1  FULL LOAD, MASTER FILE CCIDS ONLY                   *
//*           RC=4  INCREMENTAL LOAD, STANDARD FORMAT                   *
//*   NOTE:    ENSURE     THAT THE CIGIN DD POINTS TO THE SAME DATASET   *
```

```
//*           IN STEP1, STEP3, AND STEP5.                                *
//*    ------------------------------------------------------------------*
//STEP1   EXEC PGM=CIGFLOD1
//STEPLIB               DD               DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGIN    DD  DSN=&&SYNTAX,DISP=(OLD,PASS)
//CIGNDVR  DD  DSN=&&TEMP1,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB),
//             UNIT=TDISK,
//             SPACE=(CYL,(1,1)),
//             DISP=(NEW,PASS)
//CIGLOG   DD  SYSOUT=*
//CIGFDELE DD  DSN=&&TEMP1D,
//             SPACE=(CYL,(1,1)),
//             UNIT=TDISK,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB,DSORG=PS),
//             DISP=(NEW,PASS)
//CIGSNAP  DD  DSN=&&SNAP,
//             SPACE=(TRK,(1,1)),
//             DCB=(BLKSIZE=31200,LRECL=80,RECFM=FB,DSORG=PS),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//*    ------------------------------------------------------------------*
//*  STEP2: CALL ENDEVOR AND PROCESS PRINT MASTER STATEMENTS.            *
//*          THIS STEP WILL ONLY BE EXECUTED FOR INCREMENTAL LOADS.      *
//*    ------------------------------------------------------------------*
//IFSTEP2 IF  (STEP1.RC = 4 ) THEN
//* CIG AUTHORIZED LIBRARY
//* ENDEVOR AUTHLIB
//* ENDEVOR CONLIB
//STEP2   EXEC PGM=NDVRC1,PARM='CIGFLODR',REGION=0M            Z180214A
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//        DD DSN=QUAL1.QUAL2.AUTHLIB,DISP=SHR
//CONLIB  DD DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//CIGNDVR  DD  DSN=&&TEMP1,DISP=(OLD,PASS)                     Z180214A
//DDMSGS   DD  SYSOUT=*,DCB=(LRECL=133,RECFM=FB)               Z180214
//DDLIST   DD  DSN=&&SEQ2048,DISP=(NEW,DELETE),                Z180214A
//             DCB=(LRECL=2048,BLKSIZE=0,RECFM=VB,DSORG=PS),   Z180214A
//             SPACE=(CYL,(10,10)),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//CIGFEMDD DD  DSN=&&SEQ1000,DISP=(NEW,PASS),                  Z180214A
//             DCB=(LRECL=1000,BLKSIZE=0,RECFM=VB,DSORG=PS),   Z180214A
//             SPACE=(CYL,(10,10)),                            Z180214A
//             UNIT=TDISK,                                     Z180214A
//             DISP=(NEW,PASS)
//ENDIF2   ENDIF
//* ------------------------------------------------------------------*
//* STEP 3:  READ THE ENDEVOR PRINT STATEMENTS & WRITE OUT FDELETE    *
//*          AND ENDEVOR PRINT SUMMARY, MASTER, COMPONENTS STMTS.     *
//*          THIS STEP WILL ONLY BE EXECUTED FOR INCREMENTAL LOADS.   *
//*           IF NO ELEMENTS QUALIFY FOR RELOAD, RC=4.                *
//*    ------------------------------------------------------------------*
//STEP3IF  IF  (STEP2.RC < 16)  THEN
//*
//DELTEMP  EXEC PGM=IEFBR14
//DEL0     DD  DSN=&&TEMP1,DISP=(OLD,DELETE),FREE=CLOSE
//DEL1     DD  DSN=&&TEMP1D,DISP=(OLD,DELETE),FREE=CLOSE
//DEL2     DD  DSN=&&SNAP,DISP=(OLD,DELETE),FREE=CLOSE
//*
//STEP3    EXEC PGM=CIGFLOD2
```

```
//STEPLIB               DD                DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGFEMDD DD  DSN=&&SEQ1000,DISP=(OLD,DELETE)                  Z180214A
//CIGIN    DD  DSN=&&SYNTAX,DISP=(OLD,PASS)
//CIGNDVR  DD  DSN=&&TEMP1,
//             DCB=(BLKSIZE=6160,LRECL=80,RECFM=FB),
//             SPACE=(CYL,(1,1),RLSE),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//CIGFDELE DD  DSN=&&TEMP1D,
//             SPACE=(CYL,(10,10)),
//             UNIT=TDISK,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB),
//             DISP=(NEW,PASS)
//CIGSNAP  DD  DSN=&&SNAP,
//             SPACE=(TRK,(1,1)),
//             DCB=(BLKSIZE=31200,LRECL=80,RECFM=FB,DSORG=PS),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//CIGLOG   DD  SYSOUT=*
//*
//STEP3C   EXEC PGM=CIGFOPCL
//STEPLIB               DD                DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGOUT   DD  DSN=&&TEMP2,
//             DCB=(BLKSIZE=13300,LRECL=133,RECFM=FBA),
//             SPACE=(CYL,(10,10)),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//ENDIF3   ENDIF
//* ------------------------------------------------------------------*
//*  STEP4A: ALLOCATE A TEMP FILE USED FOR FULL LOADS ONLY.           *
//* ------------------------------------------------------------------*
//STEP4AIF  IF   (STEP1.RC = 0) THEN
//STEP4A   EXEC  PGM=IEFBR14
//CIGPRINT DD  DSN=&&TEMP2,
//             DCB=(BLKSIZE=13300,LRECL=133,RECFM=FBA),
//             SPACE=(CYL,(10,10)),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//SYSPRINT DD  SYSOUT=*
//ENDIF4A  ENDIF
//*    ------------------------------------------------------------------*
//*  STEP4: CALL ENDEVOR AND PROCESS PRINT STATEMENTS.                   *
//*         NOTE THIS STEP WILL BE EXECUTED FOR BOTH FULL AND         *
//*         INCREMENTAL LOADS.                                        *
//*    ------------------------------------------------------------------*
//STEP4IF   IF   (STEP1.RC = 0 | STEP3.RC < 4) THEN
//* CIG AUTHORIZED LIBRARY
//* ENDEVOR AUTHLIB
//* ENDEVOR CONLIB
//STEP4    EXEC PGM=NDVRC1,PARM='C1BM3000',REGION=0M
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//        DD DSN=QUAL1.QUAL2.AUTHLIB,DISP=SHR
//CONLIB DD DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//BSTIPT01 DD  DSN=&&TEMP1,DISP=(OLD,DELETE)
//CIGPRINT DD  DSN=&&TEMP2,DISP=(OLD,PASS)
//C1MSGS1  DD  SYSOUT=*
//ENDIF4   ENDIF
//* ------------------------------------------------------------------*
//* STEP 5A: PROCESS SNAP LABEL TAG COLLECTION. THESE WILL BE REAPPLIED
//*          AFTER THE FLOAD UPDATES HAVE OCCURRED.
//* USAGE:   NON-GREENHOUSE USERS CAN DELETE THIS STEP.
```

```
//* ------------------------------------------------------------------*
//STEP5IF  IF  (STEP4.RC < 8 | STEP3.RC < 5) THEN
//STEP5A   EXEC PGM=CIGTAG09
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGRPT   DD  SYSOUT=*
//CIGLOG   DD  SYSOUT=*
//CIGIN    DD  DSN=&&SNAP,DISP=(OLD,DELETE)
//CIGSNAP  DD  DSN=&&TAGSYN,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB),
//             SPACE=(CYL,(1,1)),
//             UNIT=TDISK,
//             DISP=(NEW,PASS,DELETE)
//* ------------------------------------------------------------------*
//* STEP 5:  PROCESS FLOAD, FDELETE, AND ENDEVOR PRINT STATEMENTS.
//*          (THE FASTLIST DATABASE WILL BE UPDATED HERE.)
//*          NOTE IF NOTHING QUALIFIED FOR THE INCREMENTAL LOAD
//*           THIS STEP WILL BE A PASS THROUGH FOR LOADS, BUT
//*           DELETES MAY STILL OCCUR.
//* ------------------------------------------------------------------*
//STEP5    EXEC PGM=CIGFLOD3
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGIN    DD  DSN=&&SYNTAX,DISP=(OLD,DELETE)
//CIGFDELE DD  DSN=&&TEMP1D,DISP=(OLD,DELETE)
//CIGINRPT DD  DSN=&&TEMP2,DISP=(OLD,DELETE)
//CIGLOG   DD  SYSOUT=*
//*CIGRPT   DD  SYSOUT=*  <== UNCOMMENT FOR DETAIL REPORT
//ENDIF5   ENDIF
//* ------------------------------------------------------------------*
//* STEP 6:  REAPPLY TAG ELEMENT SYNTAX.
//* USAGE:   NON-GREENHOUSE USERS CAN DELETE THIS STEP.
//* ------------------------------------------------------------------*
//STEP6IF  IF  (STEP5A.RC = 0) THEN
//STEP6    EXEC PGM=CIGTAG09
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGLOG   DD  SYSOUT=*
//CIGRPT   DD  SYSOUT=*
//CIGIN    DD  DSN=&&TAGSYN,DISP=(OLD,PASS)
//ENDIF6   ENDIF
//* ------------------------------------------------------------------*
//* STEPS 7 AND 8:  BUILD THE 'EE' TYPE CONTROL DATA RECORDS.
//* USAGE:   THIS STEP IS REQUIRED FOR CCID AND COMMENT VALIDATION.
//* NOTE1:   CIGDBJ15 IS THE STAND ALONE JCL FOR THESE TWO STEPS.
//* ------------------------------------------------------------------*
//* ------------------------------------------------------------------ *
//*                                                                    *
//* STEP7: PERFORM BATCH ADMIN FUNCTIONS AGAINST SELECTED ENVIRONNENTS*
//*                                                                    *
//* ------------------------------------------------------------------ *
//STEP7IF  IF  (STEP5.RC < 12) THEN
//STEP7    EXEC PGM=NDVRC1,PARM='ENBE1000',DYNAMNBR=1500,REGION=0M
//* ENDEVOR AUTHLIB
//* ENDEVOR CONLIB
//STEPLIB DD DSN=QUAL1.QUAL2.AUTHLIB,DISP=SHR
//CONLIB  DD DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//C1MSGS1   DD  SYSOUT=*
//C1MSGS2   DD  SYSOUT=*
//OUTFILE   DD  DSN=&&BATDATA,DISP=(NEW,PASS),
//          SPACE=(CYL,(10,10)),UNIT=TDISK,
//          DCB=(LRECL=80,BLKSIZE=31200,RECFM=FB)
//ENESCLIN  DD  *
 BUILD SCL FOR SYSTEM *
```

```
   FROM ENVIRONMENT 'YOUR-ENVIRONMENT-NAME'
   INCLUDE SUBORDINATES
   TO DDNAME 'OUTFILE'
 .
//STEP7END  ENDIF
//* ------------------------------------------------------------------ *
//*                                                                    *
//* STEP8: PERFORM CONTROL DATA UPDATE TO FASTLIST DATABASE.           *
//*                                                                    *
//* ------------------------------------------------------------------ *
//STEP8IF  IF  (STEP7.RC < 12) THEN
//* CIG AUTHORIZED LIBRARY
//STEP8    EXEC PGM=CIGELOAD,REGION=0M
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGIN     DD  DSN=&&BATDATA,DISP=(OLD,DELETE)
//CIGLOG    DD  SYSOUT=*
//CIGPUNCH  DD  SYSOUT=*
//CIGDB     DD  DSN=FLHQ1.FLHQ2.FLSTDB,DISP=SHR
//STEP8END  ENDIF
```

Figure 3.3
Full JCL to Run FLOAD

FLOAD Processing Outline

The FLOAD process includes several steps, many of which execute conditionally based on previous steps. This table provides a brief description of the steps in the JCL above. Additional information about specific return codes for these steps is found at the end of this chapter.

Job Step	Purpose
STEP0	Allocate two temporary files used in later steps of the process
SYNTAX	Write the FLOAD syntax specified in in-line control cards to a temporary file for use in following steps
STEP1	Parse the FLOAD syntax and determine if an incremental or full load is requested. the return code from this step signals the type of load being performed.
STEP2	This Endevor batch processing step is only executed for incremental loads. It processes the PRINT ELEMENT SCL formatted in STEP1.
STEP3	This FLOAD step is only executed for incremental loads. It examines the element print output from STEP2 and determines if the contents of Endevor and FastLIST are identical. If elements are missing or out of date in the FastLIST database then additional PRINT ELEMENT SCL statements are formatted
STEP4	This Endevor batch processing step is executed whenever elements must be loaded into the FastLIST database. It will be skipped if no elements are missing from the FastLIST database.
STEP5A	This step extracts Element Tag (also known as Greenhouse Label) information for elements that will be reloaded into the FastLIST database.
STEP5	This FLOAD step examines the element print output from STEP4 and loads the information into the FastLIST database. It also processes FDELETE statements for elements that are in FastLIST but are removed from Endevor.
STEP6	This step re-applies the Element Tag information extracted in STEP5A
STEP7	This Endevor Batch Administration step creates inventory definitions statements that describe the Endevor environment. This step is optional and can be suppressed without affecting the FLOAD process.
STEP8	This FastLIST step examines the inventory definitions created in STEP7 and inserts inventory definition records into the FastLIST database.

FLOAD Syntax

The following figure shows all of the FLOAD syntax. FLOAD syntax must follow the same order shown in figure 3.4.

FLOAD ELEMENT element [*THROUGH element*]
FROM ENVIRONMENT environment
SYSTEM system
SUBSYSTEM subsystem
TYPE type
STAGE NUM stage – number
INCREMENTAL •

Figure 3.4
FLOAD Syntax

Required Clauses

FLOAD ELEMENTS ***element-name***

Indicates the 1 -10 character element name to be loaded. May be wild carded.

FROM ENVIRONMENT *env-name*
SYSTEM system-name
SUBSYSTEM subsystem-name
TYPE type-name
STAGE NUMBER *stage-number*

The FROM parameters indicate the Endevor inventory location that will be loaded into the FastLIST database. FastLIST requires that the environment, system, subsystem, type, and stage number keywords be included for the element(s) to be loaded. Full or partial Wild carding is allowed for system, subsystem, type, and stage number.

Optional Clauses

THROUGH element-name

Use of this option limits the loading of elements to the range first element-name up to and including the through element value.

INCREMENTAL

Use of this option will cause a selective load that synchronizes the FastLIST database with Endevor. If you incrementally load an element that exists in FastLIST, but not in Endevor, it will be deleted from FastLIST. If an element exists in Endevor but not in FastLIST, then the element will be loaded. If an element exists in both Endevor and FastLIST but has changed since the last FLOAD, then the element will be loaded.

Note that incremental and full loads cannot be mixed in a single job. If multiple syntax blocks are provided and any block is a full load, then all FLOAD requests will be processed as full loads.

Syntax and Execution Report Examples

Syntax Example

Figure 3.5 shows *flhq1.flhq2.*SAMPLIB(FLOADIVP), which contains a sample of the syntax used to load an entire ENVIRONMENT into FastLIST.

```
FLOAD ELEMENTS *
     FROM ENV PROD
          SYS  *
          SUB  *
          TYPE  *
          STAGE NUM  *
           .
```

Figure 3.5
FLOADIVP

Execution Report Format

Submitting the FLOAD JCL using the syntax in figure 3.5 results in an execution report, shown in figure 3.6, printed to CIGLOG.

```
11:32:29  FST1001I  DATE 94/07/22 TIME 11:32:29  XIF/FLOAD UTILITY
11:32:29  FST1002I                E X E C U T I O N  R E P O R T
11:32:29  FST1140I  FLOAD PARSE BEGINS.
11:32:29  FST1102I  PARSER BEGINS
11:32:29  FST0020I  FLOAD ELEMENTS *
11:32:29  FST0020I       FROM ENV PROD
11:32:29  FST0020I            SYS  *
11:32:29  FST0020I            SUB  *
11:32:29  FST0020I            TYPE  *
11:32:29  FST0020I            STAGE NUM  *
11:32:29  FST0020I .
11:32:29  FST1103I  PARSER ENDS, RC=0000
11:32:29  FST1141I  FLOAD PARSER ENDS. RC=0000
```

Figure 3.6
FLOAD Execution Report

Optional Detailed Output

Allocating the CIGRPT DD directs FLOAD to print a detailed report. The format of this report is shown in figure 3.7.

```
11:33:37 FST1180I RECORD CREATED FOR CCID LB001 PARENT=$MODE/TEST/SYSA/SUBA/MAC/1/01/00
11:33:37 FST1180I RECORD CREATED FOR CCID WITHHOOK PARENT=$AMODE/TEST/SYSA/SUBA/MAC/1/01/00
11:33:37 FST1184I RECORD CREATED FOR ELEMENT $AMODE PARENT=$AMODE/TEST/SYSA/SUBA/MAC/1/01/00
11:33:38 FST1180I RECORD CREATED FOR CCID LB001 PARENT=$BSCLDS/TEST/SYSA/SUBA/MAC/1/01/00
11:33:38 FST1180I RECORD CREATED FOR CCID WITHHOOK PARENT=$BSCLDS/TEST/SYSA/SUBA/MAC/1/01/00
11:33:38 FST1184I RECORD CREATED FOR ELEMENT $BSCLDS PARENT=$BSCLDS/TEST/SYSA/SUBA/MAC/1/01/00
11:33:38 FST1180I RECORD CREATED FOR CCID LB001 PARENT=$CALL/TEST/SYSA/SUBA/MAC/1/01/00
11:33:38 FST1180I RECORD CREATED FOR CCID WITHHOOK PARENT=$CALL/TEST/SYSA/SUBA/MAC/1/01/00
11:33:38 FST1184I RECORD CREATED FOR ELEMENT $CALL
```

Figure 3.7
FLOAD Trace Report

CIGFLOAD Return Codes

In addition to flagging errors, CIGFLOAD return codes are used to control step execution during incremental and full loads. The return code explanations by job step are included below. A return code of 16 on any step will stop FLOAD processing.

Return Code	Meaning
Job Step STEP0: 00	Files were allocated correctly
	Any other return code identifies an error
Job Step SYNTAX 00	FLOAD syntax was correctly written to the temporary file
	Any other return code identifies an error
Job Step STEP1 00	All FLOAD full load requests were correctly formatted. Execution will proceed with full load processing only.
04	All FLOAD incremental load requests were correctly formatted. Execution will proceed with incremental load processing only.
12	The FLOAD syntax submitted was incorrectly formatted.
Job Step STEP2 00	Data was extracted from Endevor successfully.
04	The requests were processed with at least one warning message. FLOAD processing will proceed for the elements successfully printed.

	08	The requests were processed with at least one caution message. FLOAD processing will proceed for the elements successfully printed.
	12	The requests were processed with at least one error message. FLOAD processing will proceed for the elements successfully printed.
Job Step STEP3	00	The incremental FLOAD process successfully identified elements to be loaded from Endevor.
	04	The incremental FLOAD process found no elements missing from the FastLIST database, but may have identified some elements to be removed from the FastLIST database.
	12	The Endevor print output was not successfully translated.
Job Step STEP4	00	The Endevor print requests were processed successfully.
	04	The Endevor print requests were processed with at least one warning message. FLOAD processing will proceed for the elements successfully printed.
	08	The Endevor print requests were processed with at least one caution message. FLOAD processing will proceed for the elements successfully printed.
	12	The Endevor print requests were processed with at least one error message. FLOAD processing will proceed for the elements successfully printed.
Job Step STEP5A	00	Element Tag (Greenhouse Label) information was successfully extracted from the FastLIST database.
	04	No Element Tag (Greenhouse Label) information was found in the FastLIST database.
Job Step STEP5	00	All Endevor elements printed were successfully loaded into the FastLIST database.
	12	The Endevor print output was not correctly processed for at least one element. Examine the CIGLOG output for more information.

Job Step STEP6	00	Element Tag (Greenhouse Label) information extracted in STEP5A was correctly re-applied.
	04	At least one Element Tag (Greenhouse Label) statement received an error during execution. Examine the CIGLOG output for more information.
	12	The input tag syntax file is empty.
Job Step STEP7	00	The Endevor Batch Administration statements were executed successfully.
	12	The Endevor Batch Administration statements received errors during execution.
Job Step STEP8	00	The FastLIST Environment records were correctly inserted in the FastLIST database.
	12	The FastLIST Environment records were not successfully created. Examine the CIGLOG output for more information.

Usage Notes

Elements are processed in the same order that they are processed by Endevor.

- Users can code more than one FLOAD statement in a run. All FLOAD syntax will be parsed in STEP1 of the FLOAD JCL. All syntax will be processed together for the rest of the job. Note that INCREMENTAL and FULL loads should not be coded together. UNLESS EVERY SYNTAX BLOCK CONTAINS AN INCREMENTAL STATEMENT, FASTLIST WILL PERFORM A FULL LOAD.

- If an element in the inventory does not have a component list, it can still be added with CCID and MCF information. For instance, macros and COBOL copy books may use a **NOPROC**.

- For best results on mass loads, perform load when there is no activity against the target MCFs. Also, load the database in manageable inventory pieces. Each run of the database load will produce a detailed report of the records created, along with miscellaneous warning and information messages.

- The parameters [DO NOT] COLLECT CCID, [DO NOT] COLLECT COMPS, and COLLECT MASTERS ONLY specify what information FLOAD collects in the database. These parameters reside in the CIGINI initialization file. If the CCID or COMPS parameter is set to no (N), FLOAD will not include that information in the database. If COLLECT MASTERS ONLY is coded, the CCID value will be set to M, and all delta-level CCID information will be excluded from the database. Note that these parameters will not only limit the initial loading of the database, but will also limit the real-time update functions of the Collector.

- The inventory FILTERS coded in the CIGINI and alternate modules affect what inventory gets loaded into or deleted from the database. All or none of the elements specified in the FLOAD input may qualify for loading, based on the inventory FILTERS. The FLOAD will continue even after encountering an element that does not match the FILTER. Check the end of the summary report to see the number of records loaded and to see if any elements failed the FILTER test.

- For a non-incremental load, the FLOAD utility will delete all database records that match the ENDEVOR inventory coded in the FLOAD syntax. It is a destructive load. If any database records match the FLOAD syntax mask provided by users, then those records will be deleted, regardless of whether they exist in ENDEVOR or not.

- Using this utility does not require re-GENERATING elements in ENDEVOR.

FastLIST 12.0

Chapter 4: CIGFEXIT - Database Collection

The FastLIST Collector

Overview

The FastLIST Collector synchronizes the FastLIST database with Endevor. As a given action comes to completion, the Collector extracts information from Endevor, including the most current element status, levels, CCIDs, and components. The Collector gathers all of this information during Endevor exit processing.

The information gathered by the Collector is then indexed and stored in the FastLIST database along with CCID and component information. You have the option to enable or disable the collection of either CCIDs or components by coding these options in the CIGINI module.

If both the CCID and COMPONENT collection options are active, the Collector will collect all available data from Endevor. You must verify that all Endevor systems are enabled for CCID=YES and MONITOR=COMPONENTS in your processors so that Endevor will capture CCID and component data. The FastLIST Collector can only collect data that is produced by and stored in Endevor.

Questions and Answers

The following are some frequently asked questions about the Collector.

Are any CIG ddnames required in my Endevor JCL to activate the Collector?

Yes and no. Collector ddnames are all optional. The CIGINI ddname is required if an alternate CIGINI module is in use. Include in your JCL a CIGINI ddname which points to the dataset containing your CIGINI alternate module. The CIGTRACE ddname is required if you want the trace invoked. Include a CIGTRACE DD DUMMY to enable the trace. Note that these JCL cards can be added via the ISPF Preferences Panel if you are submitting the Endevor jobs from the FastLIST or Endevor ISPF front end.

How does the Collector know what inventory to monitor?

The active CIGINI or alternate module will determine which inventory will be monitored by the Collector. If no FILTER options are coded in the CIGINI module, all inventory will be monitored. Part of the FastLIST implementation process is to define inventory to load into the FastLIST database. This inventory should then be defined in the CIGINI module. For more information, see Chapter 3, *Implementation.*

What happens if an element is processed without CCIDs?

FastLIST will monitor and enforce the CCID options defined in the Endevor implementation. If CCIDs are required for a particular system then FastLIST prompt screens will enforce this requirement. As always, a user can find ways to bypass this requirement in which case Endevor's internal validation will cancel the action.

If your site uses CCID validation (using the standard CIPO dataset or an exit) FastLIST has no ability to participate in this validation. CCID validation will only be performed during Endevor action processing.

What happens if I do not code MONITOR=COMPONENTS?

If there are no MONITOR=COMPONENTS coded in a processor, FastLIST cannot record component information. FastLIST will still update element information in the database. However, FastLIST will not send a message or cancel an action due to lack of component information.

What happens if neither CCIDs nor components are enabled in the CIGINI module?

The exit will only collect master information if components and CCIDs are not enabled.

Can FLOAD run while the Collector is active?

Yes. In fact, you may want to perform the initial load while the Collector is active to prevent the possibility of omitting data entered between the time that the load is performed and the Collector is activated.

What happens if Endevor abends?

If Endevor abends while processing an element, you need only to examine the abend and resubmit the action. If, however, the action was processed but Endevor abended prior to EXIT 3 processing, run an incremental FLOAD to synchronize the FastLIST database.

What happens if some other EXIT 2 module cancels the action?

If another EXIT 2 application cancels the action, the Collector will not update the FastLIST database. The database will not update because the action was not executed.

Does the Collector work in batch and foreground?

Yes.

What happens if C1BM3000 is invoked in a processor?

The invocation of C1BM3000 in a processor will cause an entire Endevor application sequence to load and execute. That means that Endevor initialization, and thus FastLIST initialization, is performed on behalf of the recursive call to Endevor. The Collector supports recursive calls to Endevor.

What happens if the generate fails for an element?

The processor list is kept regardless of the generated return code. Thus FastLIST will always update the processor list. FastLIST will reflect whatever Endevor updates.

What happens on a MOVE action that does not generate the element at the target?

Again, FastLIST will reflect what Endevor has done. If the target component list was rebuilt from the source, FastLIST will rebuild the target from the source. If the element was generated, FastLIST will reflect this condition. FastLIST synchronizes the component data from the source to the target location on a MOVE without a generate.

Do I still need to execute BC1PMVCL if I am moving an element without generating?

This is your choice. FastLIST does not replace ACM data capture, rather extracts and indexes the information. If you do not update the component list after a certain point in your life cycle, then the information in FastLIST will reflect the current level component data.

What happens when to the collection if the Action abends?

When the processed the element via it's exit 2 but did not complete exit 3 processing the utility CIGFSYNC should be run to resynchronize the FastLIST records with Endevor information. CIGFSYNC is documented in the Utilities chapter of this manual.

Why do PRINT messages show up in the optional C1MSGS2 ddname after action processed by the collector?

The collector resyncs the database after every action, regardless of the action type. This is done through the use of PRINT Action statements. Endevor does not distinguish between the initial action and actions submitted in a processor or exit. Thus these internal messages can not be suppressed. They are informational only.

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:

1. Loads the CIGINI module and checks for an override CIGINI module, loading the override module if found.
2. Loads all the exit programs.
3. Performs password check.
4. Loads the application from the loadlib in the CIGINI module.
5. Allocates and opens the database defined in the CIGINI or the override module.
6. Allocates and anchors storage for duration of Endevor session.
7. Establishes FastLIST execution framework.

If an Endevor exit 5 fails, Endevor initialization fails. The following conditions will cause the FastLIST initialization to fail.

1. Invalid password.
2. Database allocation error.
3. Application loadlib allocation error.
4. Incorrect WORK or VIO value.

Note that an installation may have more than one exit 5 in its exit tables. If this is the case, then some other exit 5 may cancel Endevor initialization. If FastLIST initialization fails, FastLIST prints error messages in the CIGOUT dataset. If you have not allocated a CIGOUT ddname to your session, FastLIST sends the message to a default location. In batch, FastLIST dynamically allocates a CIGOUT ddname and assigns a SYSOUT=* attribute to the ddname. In foreground, any error messages are written to the terminal screen.

Exit 6: Delete and Terminate the Collector

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

1. De-allocate any remaining storage.
2. Close and de-allocate any files.

Exit 2: The Before Action Exit

The FastLIST before action exit collects information about the condition of an element before an action is performed, and stores this information in an anchored exit block that is then passed to exit 3.

The following conditions will cause the FastLIST exit 2 to fail.

1. CIGINI initialization module is missing.
2. FastLIST is unable to get storage for the exit block.

Note that if FastLIST exit 2 or any other Endevor exit 2 program sends back a return code higher than eight, then the action will be canceled.

Exit 3: The After Action Exit

The Collector takes the return code and MCF CCID information from exit 3 and records it in the FastLIST database. Keep in mind that as a set of Endevor exits, the Collector is not active until Endevor performs exit processing. Thus, Endevor must invoke exit 3 processing to call the Collector.

Four factors determine the amount of work done in exit 3:

1. The action, options and return codes from the action.
2. Whether there are CCIDs.
3. Whether there is a component list to process.
4. The COLLECT options set in the CIGINI file.

If a high return code is returned from exit 3, Endevor continues on to the next action. Any error messages are recorded in the CIGOUT message dd.

Collector Issues per Action

Figure 4.1 shows the information collected for each Endevor action, and the exit where that information is collected.

ACTION	Exit Point	Comments
Add	2	Collect previous version, level, processor, and source date information.
	3	Database is updated to reflect CCIDs, level activity, and current level component information.
Update	2	Collect previous version, level, processor, and source date information.
	3	Database is updated to reflect CCIDs, level activity, and current level component information.
Retrieve	2	Nothing collected.
	3	CCIDs are updated to reflect the retrieve CCID.
Delete	2	Nothing collected.
	3	All records for the deleted element are deleted or just the components are deleted.
Generate	2	Collect version, level, and processor date information.
	3	Component list updated. CCIDs will reflect updated master CCIDs. If a copyback was performed, current level information is updated.
Move	2	Collect target and source version, level, and date information.
	3	Based on options, the existence of the target element, the type of processor, the CCIDs, the levels, and the components are updated to reflect both the target and the source locations.
Transfer	2	Collect target and source version, level, and date information.
	3	Based on CIGINI Collector options, the existence of the target element, the type of processor, the CCIDs, the levels, and the components are updated to reflect both the target and the source locations.
Signin	2	Nothing collected.
	3	Key element information is updated as per action.
Restore	2	Collect target version, level, and date information.
	3	Update the database to reflect current component, CCID, and level information.
Archive	2	Determine if delete is specified.
	3	If delete specified, deletes all records associated with the archived/delete element.

Figure 4.1
Collector Issues Per Action

FastLIST 12.0

Chapter 5: Duplicate Element Exits

Duplicate Element Exit

Integrated with FastLIST is the ability to control duplicate elements that may accidentally be introduced into an incorrect Endevor inventory location. An element with the same name in an incorrect inventory location can have severe consequences. Such a mistake could result in an incorrect executable being moved into your production environment resulting in a serious production outage.

The FastLIST Duplicate Element Exit allows the presence of duplicate elements within an Endevor location to be controlled and managed. The Duplicate Element Exit is controlled through the use of a Duplicate Element Rule Set.

What is a duplicate element?

A duplicate element is defined as an element with the same name located in a different Endevor inventory location, or an element associated with more than one element type. For example, an element called PGM001A can be associated with the type COBOL and JCL.

It may be acceptable to allow duplicate element names to be associated with certain types, but unacceptable for other types. For example, you may need to allow duplicates for the types COBOL and JCL, but not allow duplicates for the type BMS (i.e., CICS maps).

How are duplicate elements controlled?

Duplicate element names are controlled with an external dataset known as the Duplicate Element Rules dataset. The syntax contained in this rules file dataset is described later in this chapter.

When is duplicate element name checking performed?

Duplicate element checking occurs during Endevor exit 2 processing. The Duplicate Element Exit rules referenced in the CIGINI file are applied to the element being added into Endevor and the FastLIST database.

The duplicate element rules are read once during the first invocation of Endevor exit 2 processing. All subsequent actions within the same session will follow the same set of rules. Consequently, the behavior of action processing will not change once an Endevor session is started.

Activating the Duplicate Element Exit

The following steps provide an overview to activating the Duplicate Element Exit facility.

Step 1: Write the Duplicate Element Exit Rules

The behavior of Duplicate Element Exit processing is controlled via rules defined to the Duplicate Element Exit Rules dataset. Duplicate Element Exit rules can be defined either as a member in a partitioned dataset or to a sequential file. The Duplicate Element Exit rules are described later in this chapter.

Once you have written the Duplicate Element Exit rules you will use the Duplicate Element Exit rules Verification Utility to check for the accuracy of the syntax.

Note: Duplicate Element Rules which are syntactically incorrect will result in the failure of all Endevor actions. Therefore, it is critical that you validate the duplicate element rules syntax prior to implementing the rules in production.

Step 2: Update the C1UEXITS (Endevor exit) table

If you have installed the FastLIST Collector or the CIG Package Utilities then there is no additional Endevor exit set-up required to install the FastLIST Duplicate Element Exit.

To check if the FastLIST Duplicate Element Exit is already set-up, check your Endevor C1DEFLTS table. The FastLIST Duplicate Element Exit is installed if the following exits have been defined in C1UEXITS table.

```
@C1UEXIT EXIT=2,NAME=CIGFEXEC,ANCHID=0,AUTH=NO
@C1UEXIT EXIT=5,NAME=CIGFEXEC,ANCHID=0,AUTH=NO
@C1UEXIT EXIT=6,NAME=CIGFEXEC,ANCHID=0,AUTH=NO
```

If the CIGFEXEC program is not defined to the C1UEXITS table then you will need to install the FastLIST exits to Endevor. To install the Endevor exits, modify the FastLIST JCL member called CIGJCL06.

Step 3: Define the Rules Dataset to the CIGINI file

Within the CIGINI file you will define the name of the dataset which contains the rules controlling the behavior of the FastLIST Duplicate Element Exit. You will add the following statement to the FASTLIST SECTION to the CIGINI file.

```
FASTLIST SECTION
     .
DUPLICATE ELEMENT RULES DATASET dsname
                          MEMBER member-name
     .
```

The MEMBER clause is required only if the file is a partitioned dataset. For sequential files, the MEMBER clause can be omitted.

Note: Once you specify the *Duplicate Element Rules Dataset* parameter to the CIGINI file the duplicate element checking will be active.

Disabling Duplicate Element Checking

There are three methods of turning off duplicate element checking.

1. You can specify the following command as the first line in the Duplicate Element Exit Rules file:

   ```
   IGNORE DUPLICATES
   ```

2. Remove the reference to the Duplicate Element Exit Rules dataset in the CIGINI file.

3. Remove the CIGFEXIT program from the Endevor C1UEXITS table.

This last method is <u>not</u> recommended since removing references to the CIGFEXIT program will deactivate the FastLIST Collector as well.

Syntax for Duplicate Element Exit Rules

The Duplicate Element Rule syntax has many of the features of a programming language. Like other languages, careful design and implementation is required to generate the desired results. Review the syntax descriptions and examples carefully before implementation.

The syntax for Duplicate Element Exit processing is classified into six categories:

- ❶ Action Control
- ❷ Logic Flow
- ❸ Current Element Checking
- ❹ Duplicate Element Checking
- ❺ Current and Duplicate Element Checking Combined
- ❻ Comments

❶ **Action Control**	These statements control Endevor actions (i.e., Add, Update, Transfer, Restore, Move).
❷ **Logic Flow**	These statements control the processing logic flow within the rules file.
❸ **Current Element Checking**	This statement allows you to identify which elements should be considered for duplicate element checking. The check is associated with the element against which the action is currently being processed.
❹ **Duplicate Element Checking**	This statement allows you to determine the processing logic that should occur in the event a duplicate element name exists within the FastLIST database.
❺ **Current and Duplicate Element Checking Combined**	This statement allows the Current Element Checking and Duplicate Element Checking to be combined into one statement (i.e., FOR clause).
❻ **Comments**	Comments can be defined to the duplicate element exit by specifying an "*" in column one of the file.

Figure 5.1
Duplicate Element Exit Syntax

Action Control Statements

Action Control statements define the end of a particular logic branch in the Duplicate Element Rules syntax. The action requested will control what return code and/or messages are communicated to Endevor and the user and will terminate processing of rules for the current action.

IGNORE DUPLICATES
ISSUE WARNING
FAIL ACTION

Syntax described

IGNORE DUPLICATES.....	This statement will cause the Duplicate Element Exit to stop its logic checking and will allow the Endevor action to proceed with an exit return code of zero.
ISSUE WARNING.....	This statement will cause the Duplicate Element Exit to stop its logic checking and will allow the Endevor action to proceed with an exit return code of a four (warning).
FAIL ACTION	This statement will cause the Duplicate Element Exit to stop its logic checking and will cancel the action with an exit return code of an eight (fail action).

Logic Flow Statements

Logic Flow statements control the branching logic of the rule set. A well-designed combination of IF statements (described below), GOTO statements, and labels can clarify the intent and implementation of rules.

label
GOTO label
EOJ

Syntax described

label..... A label is associated with a GOTO statement. It must begin in column 1 and must not be one of the following keywords: IGNORE, ISSUE, FAIL, GOTO, EOJ, IF, or FOR. Labels must be 1 to 8 characters in length.

GOTO..... This statement will cause logic processing to go to the specified label.

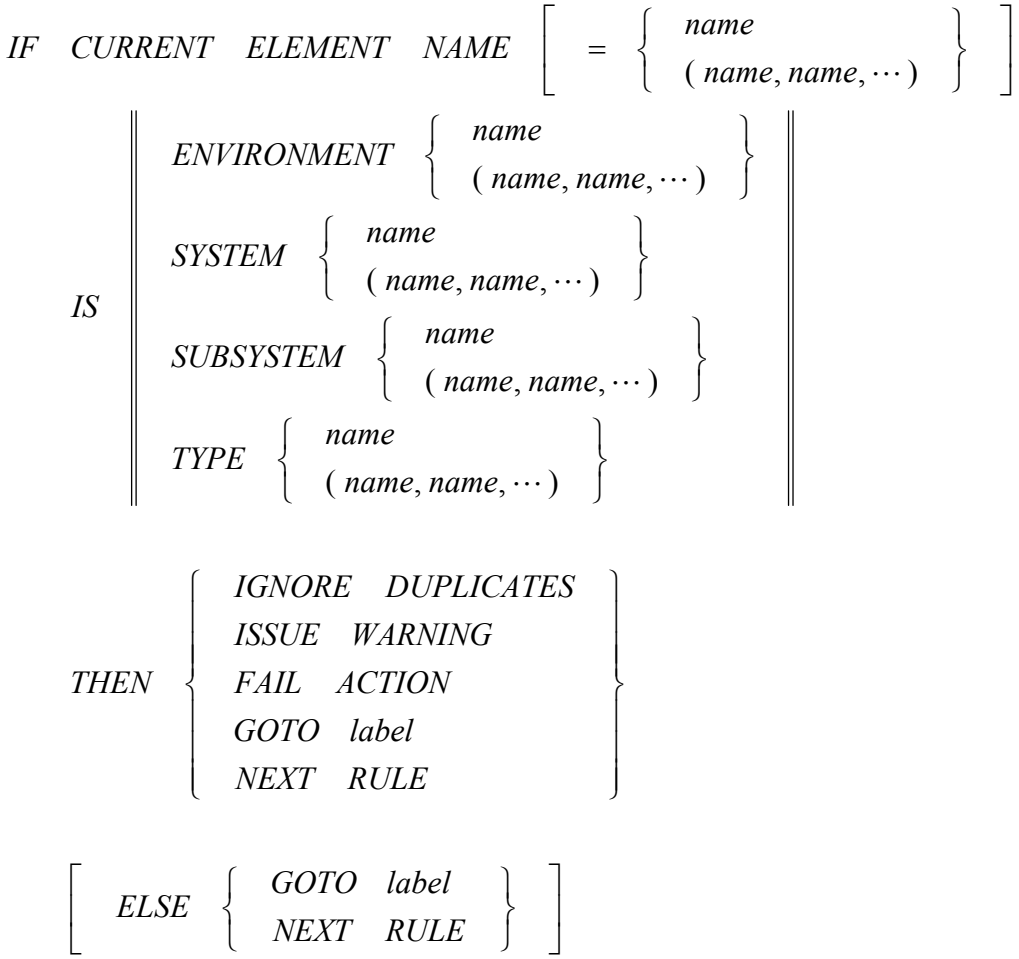
EOJ..... This statement will cause logic processing to stop immediately. When logic processing is terminated the default IGNORE DUPLICATE action processing will be executed.

Use the EOJ statement or an Action Control statement at the end of every block of logic to ensure that processing doesn't accidentally "fall through" with unintended consequences.

Current Element Checking

Current Element checking is the primary method for controlling logic flow in the rule set. Using IF statements carefully can clearly steer processing in a clear and efficient manner. Do not attempt to check all conditions with a single statement. Work in a top-down manner and utilize GOTO statements to break complex conditions into logical blocks.

Note that Current Element checking uses only information contained in the exit blocks for the current action. No searching of the FastLIST database is done during the processing of this statement.



Syntax described

IF CURRENT ELEMENT ..... This statement will cause the duplicate element exit to check the location of the element against which an action is being performed.

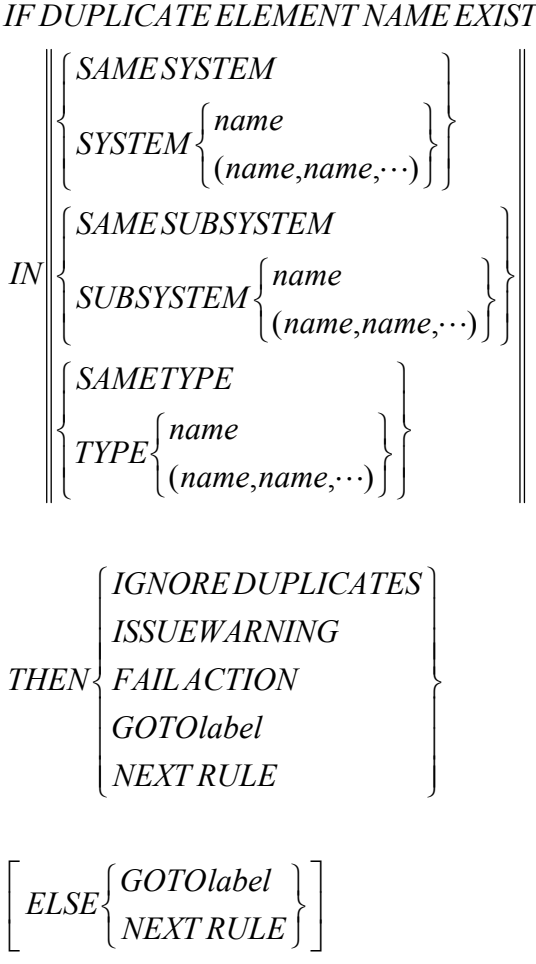
IS *clause* ..... This operand allows current element checking to be restricted to a specific Endevor inventory location.

THEN	This operand gets executed if the IF directive tests true. If the IF directive tests false then the ELSE clause will be executed, if present.
IGNORE DUPLICATES, ISSUE WARNING, FAIL ACTION	See Action Control Processing for an explanation of these options.
GOTO *label*	Processing control will go to the label associated with this option.
NEXT RULE.....	Processing control will execute the next rule in this file.
ELSE	This operand gets executed if the IF directive tests false. If no ELSE directive is coded then the next statement in the rules file will be executed.
GOTO *label*...	Processing control will go to the label associated with this option.
NEXT RULE	Processing control will execute the next rule in this file.

Duplicate Element Checking

Duplicate Element checking implements the comparison of the current element with the content of the FastLIST database. As with the Current Element checking, use simple rules and break the rules into logical blocks to keep processing efficient and increase maintainability.

Note that processing of this rule requires accessing of the FastLIST database. Minimize the use of this rule if the Action Control result is IGNORE.



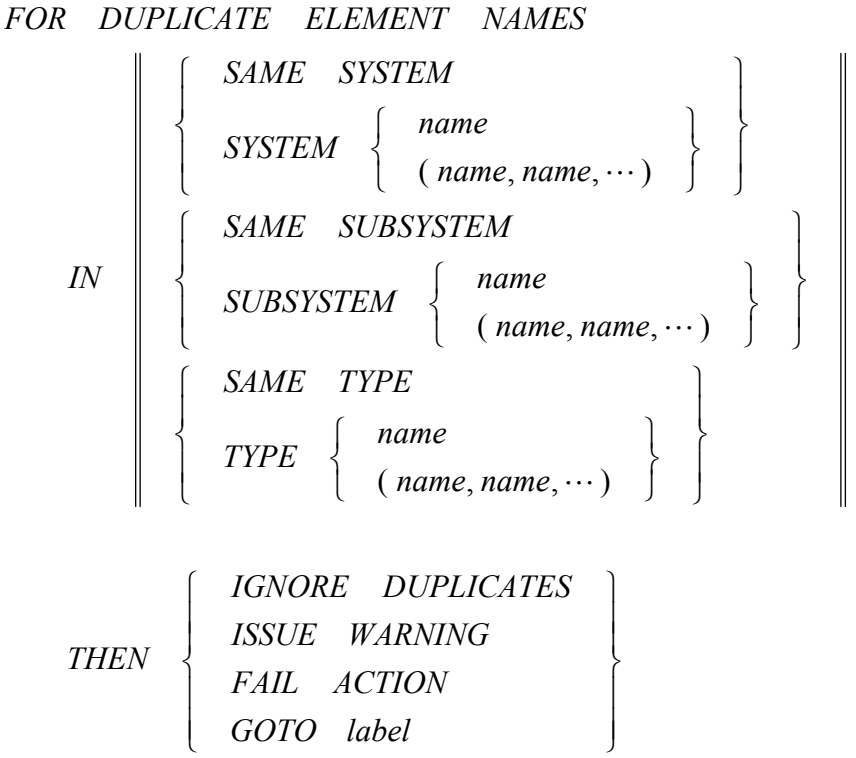
Syntax described

IF DUPLICATE ELEMENT NAME EXISTS	This check will cause the Duplicate Element Exit to check for elements contained in the FastLIST database.

SAME.....	This directive causes the check to be applied to the same inventory location as the element against which the element action is being performed.
THEN	This operand gets executed if the IF directive tests true. If the IF directive tests false then the ELSE clause will be executed, if present. If no IF directive is coded then the next statement in the rules file will be executed.
IGNORE DUPLICATES, ISSUE WARNING, FAIL ACTION	See Action Control Processing for an explanation of these options.
GOTO *label*	Processing control will go to the label associated with this option.
NEXT RULE.....	Processing control will execute the next rule in this file.
ELSE	This operand gets executed if the IF directive tests false. If no ELSE directive is coded then the next statement in the rules file will be executed.
GOTO *label*	Processing control will go to the label associated with this option.
NEXT RULE.....	Processing control will execute the next rule in this file.
(*name, name, . . .*)	A list of names can be specified. If specified then the condition tests true if any of the names in the list match the inventory location of the duplicate element location.

Current and Duplicate Element Checking Combined

Current and Duplicate Element Checking rules provide a shortened notation for rule checking. In many cases one or a few FOR clauses will implement all logic required for a site. As with the Duplicate Element rule above, the FastLIST database is accessed to resolve this rule.



Syntax described

FOR DUPLICATE NAMES IN — This rule combines the Duplicate Element Checking with the Current Element Checking. The specified element must both be the current element being processed and match an existing duplicate element name in the FastLIST database.

SAME..... — This directive causes the check to be applied to the same inventory location against which the element action is being performed.

THEN ..... — This operand gets executed if the IF directive tests true.

IGNORE DUPLICATES, — See Action Control Processing for an explanation of

ISSUE WARNING, FAIL ACTION	these options.
GOTO *label*...	Processing control will go to the label associated with this option.
(*name, name, . . .*)	A list of names can be specified. If specified then the condition tests true if any of the names in the list inventory location of the duplicate element location.

Rules Examples

The following illustrates the rules previously described:

```
* THIS IS AN EXAMPLE OF DUPLICATE ELEMENT RULES.
* RULE #1
   IF CURRENT ELEMENT NAME IS SYSTEM FINANCE
     THEN GOTO FINCHECK
* RULE #2
   IF CURRENT ELEMENT IS TYPE (BMS, COB*)
     THEN GOTO BMSCHECK
* RULE #3
   IF DUPLICATE ELEMENT NAMES EXIST
          IN SAME SYSTEM SAME SUBSYSTEM
     THEN FAIL ACTION
* RULE #4
   IGNORE DUPLICATE


* RULE #5
FINCHECK
* RULE #6
IF DUPLICATE ELEMENT NAME EXISTS
          IN SYSTEM (FINANCE, HR, PAYROLL)
     THEN ISSUE WARNING
* RULE #7
   IGNORE DUPLICATE

* RULE #8
BMSCHECK
* RULE #9
FOR DUPLICATE ELEMENT NAMES IN SYSTEM ADMIN
     THEN ISSUE WARNING
* RULE #10
IGNORE DUPLICATES

* RULE #11
EOJ
```

Syntax described

Rule 1 ..... This rule checks if the current action is being performed in the system FINANCE. If true then processing logic will proceed to the label FINCHECK. If false then processing logic will proceed to rule 2.

Rule 2 ..... This rule checks if the current element is type BMS or if the type name starts with the characters COB. If true, then processing will continue at the BMSCHECK label. If false, then processing will continue with rule #3.

Rule 3 ..... This check will compare the elements in the FastLIST database against the current element. If the element name being processed matches an element name is in the same system and subsystem (and a different type) then the action will be canceled. If the element is in a different system or subsystem then duplicate element checking will continue with rule 4.

Rule 4 ..... This is a default condition which is invoked when rules 1, 2, and 3 test false. By default, duplicate elements will be allowed (IGNORE DUPLICATES).

Rule 5 ..... FINCHECK is a label.

Rule 6 ..... This rule gets processed when rule 1 tests true. Control is transferred to this rule via the GOTO FINCHECK statement in rule 1. In rule 6, a warning message will be issued if the element being added into Endevor already exists in any of the following systems: FINANCE, HR, PAYROLL.

Rule 7 ..... This is the default rule associated with the FINCHECK label. If the current element name is not duplicated in systems FINANCE, HR or PAYROLL then the rules will exit with no further action.

Rule 8 ..... BMSCHECK is a label.

Rule 9 ..... A duplicate element check will be performed against element names in the ADMIN system. If a duplicate element name is identified in any of the subsystems within ADMIN then a warning message will be issued.

Rule 10 ..... This is the default rule associated with the BMSCHECK label. If the current element name is not duplicated in the ADMIN system then the rules will exit with no further action.

Rule 11 ..... The EOJ causes duplicate element checking to stop.

Duplicate Element Exit rules syntax can be verified with the Verification Utility. This utility should be executed prior to moving the duplicate element rules into production.

JCL

```
//* JOBCARD
//CIGFDUPL   EXEC   PGM=CIGFDUPL
//STEPLIB    DD  DSN=product.loadlib,DISP=SHR
//CIGIN      DD  *
     syntax rules go here
//CIGLOG     DD  SYSOUT=*
```

Ensure your duplicate element exit rules syntax is valid. Invalid syntax defined to the Rules dataset will result in all Endevor actions being canceled.

FastLIST 12.0

Chapter 6: Utilities

Summary of Batch Utilities

Control File Utilities

• Compile a CIGINI module	CIGJCL04
• Define the FastLIST Collector to the Endevor table	CIGJCL06
• Validate syntax of Duplicate Element Rules File	CIGJCL14

Database Maintenance Utilities

• Allocate a FastLIST database	CIGDBJ02
• Perform an incremental or full database load (FLOAD utility)	CIGJCL07
• Validate syntax of Duplicate Element Rules File	CIGJCL14
• Perform a full database load (FLOAD utility non-Greenhouse users)	CIGJCLF7
• Backup, restore and reorganize the FastLIST database	CIGDBJ04
• Delete an element definition from the FastLIST database	CIGJCL10
• Add “EE” control records to FastLIST database	CIGDBJ15
• Synchronize FastLIST records after action ABEND	CIGFSYNC

Allocate a FastLIST database (CIGDBJ02)

The FastLIST database is a standard VSAM KSDS file and is allocated using the IDCAMS VSAM utility.

The CIGDBJ02 JCL member is a two-step job. In Step 1, a control record is written to a temporary file. In Step 2 the new database is allocated and the control record is copied into the newly allocated database. The JCL member CIGDBJ02 should be modified and submitted to allocate your FastLIST database.

You will need to modify the DEFINE CLUSTER specification in Step 2 to reflect your site naming standards as well as DASD requirements for the FastLIST database. The following parameters will need to be modified:

- `NAME` Specifies the name of the FastLIST database
- `TRACKS` Specifies the amount of space to be allocated. Cylinder allocation is supported and recommended.
- `VOLUMES` Specifies the VOLSER where the dataset will be placed

Space Requirements

Refer to chapter two of this manual for a discussion of calculating the size of a FastLIST database.

Member: CIGDBJ02
Location: JCL Library

```
//**(JOBCARD)
//**
//*********************************************************************
//*                                                                   *
//* CIGDBJ02 - THE PURPOSE OF THIS JCL IS TO ALLOCATE A FASTLIST      *
//*            VSAM DATABASE. THE SPACE ALLOCATIONS WERE BASED ON     *
//*            AN INVENTORY OF 1200 PROGRAMS AND 45 MACROS.           *
//*                                                                   *
//*********************************************************************
//*                                                                   *
//* MODIFY THIS JCL TO MEET YOUR SITE REQUIREMENTS IN THE FOLLOWING   *
//* WAYS.                                                             *
//*                                                                   *
//*            1) INCLUDE A JOBCARD                                   *
//*            2) CHANGE FLHQ1 AND FLHQ2 AS PER WORKSHEET.            *
//*            3) CHANGE DVOLSER TO THE VOL SER OF THE DISK USED TO   *
//*               STORE THE FASTLIST DATABASE. YOU MAY NOT NEED THIS  *
//*               PARAMETER IF VSAM ALLOCATION IS MANAGED BY SMS.     *
//*            4) CHANGE ALL REFERENCES TO 'FLHQ1.FASTLIST.DATABASE'  *
//*               TO THE NAME OF YOUR FASTLIST DATABASE.              *
//*            5) CHANGE TDISK TO REFLECT THE WORKSHEET VALUE         *
//*               FOR TEMPORARY DISK UNIT.                            *
//*********************************************************************
//*                                                                   *
//* DO NOT MODIFY THE VSAM PARAMETER PROVIDED IN THIS JCL. DOING SO   *
//* WILL PRODUCE UNEXPECTED RESULTS FROM THE FASTLIST                 *
//* APPLICATION.                                                      *
//*                                                                   *
//*********************************************************************
//STEP1    EXEC PGM=CIGFPOPL
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGOUT   DD  DSN=&&TEMP,DISP=(NEW,PASS),
//             SPACE=(1,1),UNIT=TDISK,
//             DCB=(BLKSIZE=187,LRECL=187,RECFM=FB)
//CIGLOG   DD  SYSOUT=*
//STEP2   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INDD01   DD  DSN=&&TEMP,DISP=(OLD,DELETE)
//SYSIN   DD *
 DELETE FLHQ1.FASTLIST.DATABASE
 DEFINE CLUSTER -
    (NAME('FLHQ1.FASTLIST.DATABASE') -
          SPEED UNIQUE FREESPACE(30 30) -
    TRACKS(60 40) -
    VOLUMES(DVOLSER) -
    SHR(4 3) -
    KEYS(72 0)  -
    RECORDSIZE(100 2400)) -
  DATA  (CISZ(16000)) -
  INDEX (CISZ(4096))
 REPRO INFILE(INDD01) OUTDATASET('FLHQ1.FASTLIST.DATABASE')
/*
```

Compile a CIGINI module (CIGJCL04)

To activate FastLIST, a control file known as the CIGINI file must be created. Information such as product password, load library location, temporary storage unit definition information, as well as the name of the FastLIST database, can be found in the CIGINI file.

The CIGINI is created through a three step process:

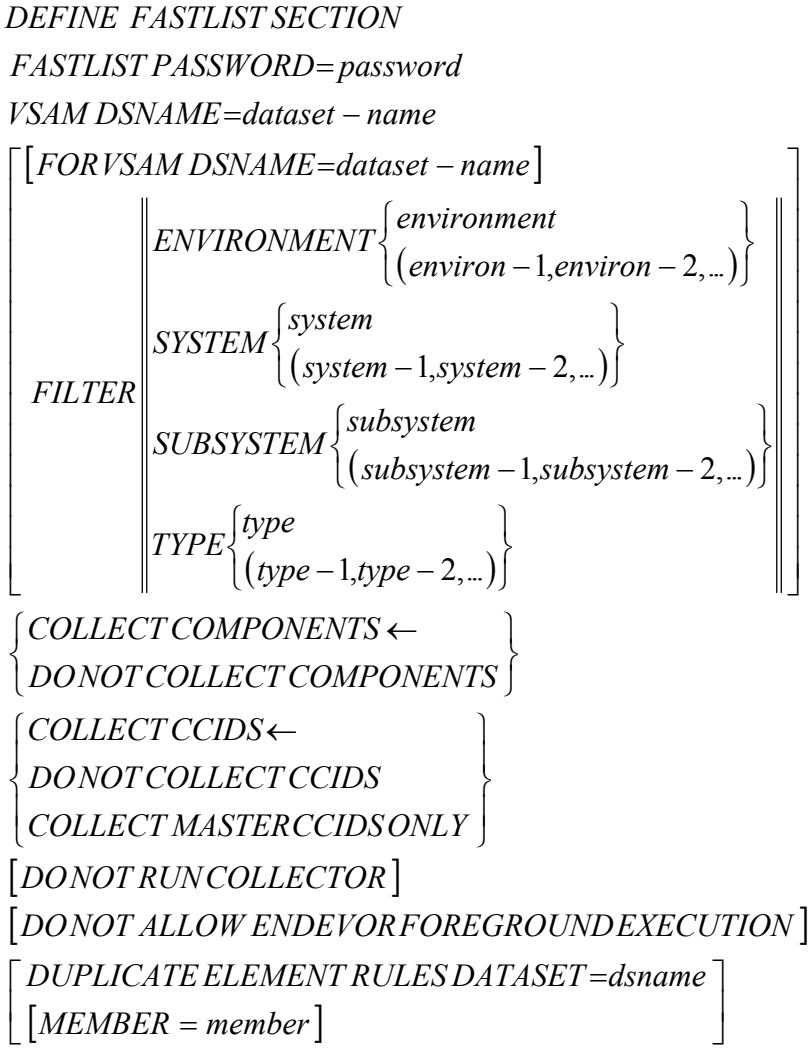
Step 1: Compile and validate the CIGINI syntax.
Step 2: Assemble the output created via Step 1.
Step 3: Create the CIGINI load module.

Ensure that all Chicago Interface Group products are properly defined to the CIGINI file before replacing any current version of the CIGINI file. The CIGINI file must reside in an authorized library and is normally placed in a LINKLIB along with the load module CIGFEXEC.

Syntax

The following describes the syntax required to activate FastLIST. The syntax is defined to the PGM=ICOMPILE via Step 1 through the CIGIN ddname.

DEFINE COMMON SECTION
PRODUCT LOADLIB = dataset – name
WORK UNIT = unit
VIO UNIT = unit
ENDEVOR CONLIB DSNAME = dataset – name
[*DO NOT ALLOW ALTERNATE CIGINI FILE*]
[*LOCK FILE VIA* { *ENQUEUE* ← / *RESERVE* } *MACRO*]
[*LINKAGE EDITOR NAME = name*]


DEFINE FASTLIST SECTION
FASTLIST PASSWORD=password
VSAM DSNAME=dataset-name
[FOR VSAM DSNAME=dataset-name]
FILTER
ENVIRONMENT {environment | (environ-1,environ-2,...)}
SYSTEM {system | (system-1,system-2,...)}
SUBSYSTEM {subsystem | (subsystem-1,subsystem-2,...)}
TYPE {type | (type-1,type-2,...)}
{COLLECT COMPONENTS ← | DO NOT COLLECT COMPONENTS}
{COLLECT CCIDS ← | DO NOT COLLECT CCIDS | COLLECT MASTER CCIDS ONLY}
[DO NOT RUN COLLECTOR]
[DO NOT ALLOW ENDEVOR FOREGROUND EXECUTION]
[DUPLICATE ELEMENT RULES DATASET=dsname
[MEMBER = member]]

DEFINE COMMON SECTION

PRODUCT LOAD LIBRARY:	Required. This is the name of the FastLIST product load library. Quotes around the dataset name are required.
WORK UNIT:	Required. Specifies the UNIT= parameter which FastLIST uses when performing dynamic dataset allocation. This parameter must be 1-8 characters in size, and is normally specified as SYSDA.
VIO UNIT:	Required. Same as WORK UNIT parameter.
ENDEVOR CONLIB DSNAME:	Required. This is the name of the dataset which contains Endevor //CONLIB members. Quotes around the dataset name are required.

DEFINE FASTLIST SECTION

FASTLIST PASSWORD:	Required. This is the FastLIST product password.
VSAM DSNAME:	Required. This is the dataset name of the primary FastLIST database. Unless a FILTER clause or FOR clause is specified, all elements loaded into the FastLIST database will be stored in the dataset specified in this parameter.
FOR VSAM DSNAME:	Optional. This is the dataset name of an alternate FastLIST database. If a FOR VSAM DSNAME statement is coded then a FILTER clause must also be specified.
FILTER:	This clause allows Endevor inventory locations to be routed to alternate FastLIST databases. If the FOR VSAM DSNAME is specified then this clause is required. If no FOR VSAM DSNAME is specified then this clause is optional, and if specified applies to the primary FastLIST database (see VSAM DSNAME above).
[DO NOT] COLLECT COMPONENTS:	This clause determines if Endevor-ACM components will be stored in the FastLIST database. By default, Endevor-ACM components will be stored in the FastLIST database.

`[ DO NOT ] COLLECT CCIDS:` This clause determines if CCID information will be stored in the FastLIST database. By default, Endevor CCIDs will be stored in the FastLIST database.

`COLLECT MASTER CCIDS ONLY:` This clause will cause only CCIDs associated with the current element level to be stored in the FastLIST database. By default, all CCIDs will be stored in the FastLIST database.

`DO NOT RUN COLLECTOR:` This clause will disable the FastLIST Collector. By turning off the FastLIST Collector, no FastLIST database updates will occur during Endevor exit processing. The FastLIST database can still be updated using the FLOAD utility.

DO NOT ALLOW ENDEVOR FOREGROUND EXECUTION:

This clause will prevent any Endevor action from being executed via the FastLIST ISPF front-end.

DUPLICATE ELEMENT RULES DATASET:

This clause will enable the Duplicate Element Prevention Facility. If the dataset is partitioned then a MEMBER name must be specified.

Model JCL

Member: CIGJCL04
Location: JCL Library

```
//**(JOBCARD)
//*--------------------------------------------------------------------*
//*                                                                    *
//* NAME:PARSE, ASSEMBLE AND LINK CIGINI MODULES                       *
//*                                                                    *
//* MODIFY THE JCL IN THE FOLLOWING WAYS.                              *
//*            1) ADD A JOB CARD                                       *
//*            2) THE SYSLMOD DATASET MUST POINT TO YOUR LOADLIB,      *
//*               CIGINI OVERRIDE DATASET, OR THE ACTUAL STEPLIB/      *
//*               LINKLST DATASET THAT WILL CONTAIN FASTLIST REQUIRED  *
//*               MODULES.                                             *
//*               CIGINI OVERRIDES ARE OPTIONAL.                       *
//*            3) CHANGE FLHQ1 FLHQ2 AND DUNIT AS PER YOUR             *
//*               INSTALLATION WORKSHEET.                              *
//*--------------------------------------------------------------------*
//*--------------------------------------------------------------------*
//*                                                                    *
//* STEP 1: PARSE CIGINI SYNTAX.  BUILD INPUT FOR ASSEMBLER.           *
//*                                                                    *
//*--------------------------------------------------------------------*
//PARSE    EXEC PGM=ICOMPILE
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGIN    DD  DSN=FLHQ1.FLHQ2.SAMPLIB(CIGINI),DISP=SHR
//CIGPUNCH 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 ALTERNATE CIGINI MODULE, YOU MUST      *
//*        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)

//SYSUT1   DD  UNIT=DUNIT,SPACE=(TRK,(5,15))
```

Update the Endevor (C1UEXITS) table (CIGJCL06)

To activate the FastLIST Collector, the program CIGFEXEC must be defined to Endevor's C1UEXITS table.

Your installation may already be running other Endevor exits. In this event you should add the program CIGFEXEC to your existing C1UEXITS table.

C1UEXITS syntax

C1UEXITS syntax is defined by coding assembler macros known as @C1UEXIT. The input file is defined to the assembler via the SYSIN dd statement. The following shows the exits that must be specified to activate the FastLIST Collector. Consult the Endevor documentation for additional documentation on the Endevor exits table.

```
C1UEXITS @C1UEXIT TYPE=START,XIT7BAT=YES
         @C1UEXIT EXIT#=2,NAME=CIGFEXEC,ANCHID=0,AUTH=NO
         @C1UEXIT EXIT#=3,NAME=CIGFEXEC,ANCHID=0,AUTH=NO
         @C1UEXIT EXIT#=5,NAME=CIGFEXEC,ANCHID=0,AUTH=NO
         @C1UEXIT EXIT#=6,NAME=CIGFEXEC,ANCHID=0,AUTH=NO
         @C1UEXIT TYPE=END
         END
```

ANCHID= This parameter is required by the @C1UEXIT macro. The FastLIST Collector does not use the ANCHID value during exit processing.

AUTH= This parameter is required by the @C1UEXIT macro. The AUTH= parameter indicates whether the program CIGFEXEC is stored in an authorized library. If AUTH=YES is specified then the program CIGFEXEC must reside in an authorized library. If AUTH=NO is specified then the program CIGFEXEC can be stored in either the STEPLIB or CONLIB ddname.

You must define CIGFEXEC to exit 7 within the C1UEXITS table if you are running the CIG Package Utilities or Greenhouse.

Model JCL

Member: CIGJCL06
Location: JCL Library

```
//**(JOBCARD)
//**
//*--------------------------------------------------------------------*
//*                                                                    *
//* NAME: CIGJCL06 - BUILT THE C1UEXITS MODULE                         *
//*                                                                    *
//* THE PURPOSE OF THIS JCL IS TO BUILD A C1UEXITS TABLE WHICH         *
//* INCLUDES THE FASTLIST COLLECTOR MODULES. PLEASE SEE THE ENDEVOR    *
//* 3.6 ADMINISTRATORS GUIDE OR THE Endevor 3.7 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.    *
//*            5) MODIFY UNIT=TDISK TO REFLECT YOUR UNIT= PARAMETER    *
//*               SPECIFICATION FOR TEMPORARY DATASET ALLOCATION.      *
//*--------------------------------------------------------------------*
//ASM      EXEC PGM=IEV90,
//             REGION=3072K,
//             PARM='NODECK,OBJECT,NOTERM,LIST,XREF(SHORT)'
//SYSIN    DD  DISP=SHR,DSN=FLHQ1.FLHQ2.SAMPLIB(C1UEXITS)
//SYSLIB   DD  DISP=SHR,DSN=QUAL1.QUAL2.SOURCE
//SYSLIN   DD  DSN=&&SYSLIN,
//             UNIT=TDISK,
//             SPACE=(TRK,(3,5)),
//             DISP=(NEW,PASS,DELETE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPUNCH DD  DUMMY
//SYSUT1   DD  UNIT=TDISK,SPACE=(TRK,(5,15))
//SYSPRINT DD  SYSOUT=*
//*--------------------------------------------------------------------*
//*                                                                    *
//* STEP 2: LINK EDIT THE C1UEXITS TABLE.                              *
//*                                                                    *
//*  NOTE THIS TABLE HAS TO BE 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  DISP=SHR,DSN=QUAL1.QUAL2.LOADLIB(C1UEXITS)
//SYSUT1   DD  UNIT=TDISK,SPACE=(TRK,(5,15))
```

Incremental database load — FLOAD (CIGJCL07)

The CIGJCL07 is used to load the FastLIST database on an incremental basis. Chapter 3 describes the incremental database loading process in detail.

Syntax

FLOAD ELEMENT element [*THROUGH element*]
FROM ENVIRONMENT environment
SYSTEM system
SUBSYSTEM subsystem
TYPE type
STAGE NUM stage – number
INCREMENTAL •

FLOAD ELEMENT:	Required. Identifies the element range to be loaded into the FastLIST database. Wild-carding is allowed.
FROM:	Required. The inventory location can be wild-carded using an "*". For example, specifying SYSTEM * will cause all systems to be loaded into the FastLIST database.
INCREMENTAL:	Required. Specifies that an incremental database load is to be performed. Omitting the INCREMENTAL clause will cause a full database load to be performed.
Period:	Required. The period terminates the FLOAD statement.

Job steps described

Step 1:	Process FLOAD syntax and create appropriate Endevor PRINT statements used by the other steps in the FLOAD jobstream.
Step 2:	Invoke Endevor to process PRINT ELEMENT statements created in Step 1 above.
Step 3:	Compare the Endevor output created in Step 2 against data stored in the FastLIST database. If Endevor elements are found which do not match FastLIST database elements then additional PRINT ELEMENT statements will be created. In additional, FDELETE statements will be generated for those Endevor elements which exist in the FastLIST database, but not exist in the Endevor Master Control File (MCF).
Step 4:	Invoke Endevor a second time to process PRINT ELEMENT statements created in Step 2 above. Only those elements which differ between the FastLIST database and the Endevor MCF will be processed.
Step 5a:	For GREENHOUSE users only, collect all labels prior to rebuilding.
Step 5:	Update the FastLIST database based on the FDELETE statements generated in Step 3 and Endevor output created in Step 4.
Step 6:	For GREENHOUSE users only, reapply GREENHOUSE labels.
Step 7:	Collect Endevor inventory definitions for the FastLIST database.
Step 8:	Load Endevor inventory definitions to the FastLIST database.

Due to an Endevor deficiency, Step 7 *must* be performed once per environment.

Model JCL

Member: CIGJCL07
Location: JCL Library

```
//**(JOBCARD)
//**
//* ---------------------------------------------------------------------*
//*  NAME:    CIGJCL07                                                   *
//*  PURPOSE: PERFORM LOAD OR INCREMENTAL LOAD OF FASTLIST DATABASE.     *
//* ---------------------------------------------------------------------*
//*  TO USE THIS JCL, YOU MUST:                                          *
//*         1) INSERT A VALID JOB CARD WITH VALID CLASS AND REGION=0M    *
//*         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 YOUR ENDEVOR             *
//*            CONLIB DATASET                                            *
//*         4) CHANGE FLHQ1, FLHQ2, QUAL1 AND QUAL2                      *
//*            AS PER YOUR INSTALLATION WORKSHEET                        *
//*         5) CHANGE THE UNIT=TDISK TO THE APPROPRIATE UNIT             *
//*            NAME.                                                     *
//*         6) REVIEW SYNTAX IN STEP SYNTAX.  MODIFY TO REFLECT THE      *
//*            TYPE OF LOAD (ADD INCREMENTAL STMT BEFORE '.') AND THE    *
//*            INVENTORY TO BE LOADED                                    *
//*         7) MAKE SURE THAT IF YOU ARE USING AN OPTIONAL CIGINI        *
//*            DD THAT YOU INCLUDE THE DD STATEMENT IN EACH OF THE       *
//*            5 STEPS.  SEE THE MANUAL FOR OTHER REQUIRED DDNAMES       *
//*         8) FOR NON-GREENHOUSE USERS, DELETE THE CIGSNAP DDNAMES,     *
//*            FROM STEPS STEP1 AND STEP3.  REMOVE JOB STEPS STEP5A      *
//*            AND STEP6.  DO NOT REMOVE ANY IF OR ENDIF STATEMENTS.     *
//*         9) MODIFY THE BUILD SCL REQUEST IN STEP7 TO REFLECT THE      *
//*            SCOPE OF YOUR FASTLIST INSTALLATION.  OPTIONALLY,         *
//*            THESE STEPS CAN BE REMOVED FROM THE LOAD JOB AND RUN      *
//*            SEPARATELY.  JCL TO EXECUTE THESE STEPS IS IN MEMBER      *
//*            CIGDBJ15 IN THE CIG PRODUCTS JCLLIB.                      *
//*        10) CHANGE THE FASTLIST DATABASE NAME IN STEP 8 TO THE        *
//*            DATABASE TO BE UPDATED.                                   *
//* ---------------------------------------------------------------------*
//* ---------------------------------------------------------------------*
//*  SYNTAX: ALLOCATE AND POPULATE THE SYNTAX DD HERE.                   *
//* ---------------------------------------------------------------------*
//SYNTAX   EXEC PGM=LISTFILE
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//SYSIN    DD  *
FLOAD ELEMENTS *
 FROM ENV YOUR-ENVIRONMENT-NAME
      SYSTEM     *
      SUBSYSTEM  *
      TYPE  *
      STAGE NUMBER  *
.
//SYSOUT   DD  DSN=&&SYNTAX,
//             SPACE=(TRK,(1,1)),
//             DCB=(BLKSIZE=31200,LRECL=80,RECFM=FB,DSORG=PS),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//* ---------------------------------------------------------------------*
//*  STEP1:  READ FLOAD STATEMENTS AND WRITE ENDEVOR PRINT STATEMENTS    *
//*          RC=0  STANDARD FULL LOAD                                    *
//*          RC=1  FULL LOAD, MASTER FILE CCIDS ONLY                     *
```

```
//*          RC=4  INCREMENTAL LOAD, STANDARD FORMAT                  *
//*  NOTE:   ENSURE    THAT THE CIGIN DD POINTS TO THE SAME DATASET    *
//*          IN STEP1, STEP3, AND STEP5.                              *
//* ------------------------------------------------------------------*
//STEP1   EXEC PGM=CIGFLOD1
//* CIG AUTHORIZED LIBRARY
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGIN    DD  DSN=&&SYNTAX,DISP=(OLD,PASS)
//CIGNDVR  DD  DSN=&&TEMP1,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB),
//             UNIT=TDISK,
//             SPACE=(CYL,(1,1)),
//             DISP=(NEW,PASS)
//CIGLOG   DD  SYSOUT=*
//CIGFDELE DD  DSN=&&TEMP1D,
//             SPACE=(CYL,(1,1)),
//             UNIT=TDISK,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB,DSORG=PS),
//             DISP=(NEW,PASS)
//CIGSNAP  DD  DSN=&&SNAP,
//             SPACE=(TRK,(1,1)),
//             DCB=(BLKSIZE=31200,LRECL=80,RECFM=FB,DSORG=PS),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//* ------------------------------------------------------------------*
//*  STEP2: CALL ENDEVOR AND PROCESS PRINT MASTER STATEMENTS.         *
//*         THIS STEP WILL ONLY BE EXECUTED FOR INCREMENTAL LOADS.    *
//* ------------------------------------------------------------------*
//IFSTEP2 IF  (STEP1.RC = 4 ) THEN
//* ENDEVOR AUTHLIB
//* CIG AUTHORIZED LIBRARY
//* ENDEVOR CONLIB
//STEP2   EXEC PGM=NDVRC1,PARM='C1BM3000',REGION=0M
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//        DD DSN=QUAL1.QUAL2.AUTHLIB,DISP=SHR
//CONLIB  DD DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//BSTIPT01 DD  DSN=&&TEMP1,DISP=(OLD,PASS)
//CIGPRINT DD  DSN=&&TEMP2,
//             DCB=(BLKSIZE=13300,LRECL=133,RECFM=FBA),
//             SPACE=(CYL,(10,10)),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//C1MSGS1  DD  SYSOUT=*
//ENDIF2   ENDIF
//* ------------------------------------------------------------------*
//* STEP 3:  READ THE ENDEVOR PRINT STATEMENTS & WRITE OUT FDELETE    *
//*          AND ENDEVOR PRINT SUMMARY, MASTER, COMPONENTS STMTS.     *
//*          THIS STEP WILL ONLY BE EXECUTED FOR INCREMENTAL LOADS.   *
//*          IF NO ELEMENTS QUALIFY FOR RELOAD, RC=4.                 *
//* ------------------------------------------------------------------*
//STEP3IF  IF  (STEP2.RC < 16)  THEN
//*
//DELTEMP  EXEC PGM=IEFBR14
//DEL0     DD  DSN=&&TEMP1,DISP=(OLD,DELETE),FREE=CLOSE
//DEL1     DD  DSN=&&TEMP1D,DISP=(OLD,DELETE),FREE=CLOSE
//DEL2     DD  DSN=&&SNAP,DISP=(OLD,DELETE),FREE=CLOSE
//*
//STEP3    EXEC PGM=CIGFLOD2
//* CIG AUTHORIZED LIBRARY
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGINRPT DD  DSN=&&TEMP2,DISP=(OLD,DELETE),FREE=CLOSE
//CIGIN    DD  DSN=&&SYNTAX,DISP=(OLD,PASS)
//CIGNDVR  DD  DSN=&&TEMP1,
```

```
//             DCB=(BLKSIZE=6160,LRECL=80,RECFM=FB),
//             SPACE=(CYL,(1,1),RLSE),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//CIGFDELE DD  DSN=&&TEMP1D,
//             SPACE=(CYL,(10,10)),
//             UNIT=TDISK,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB),
//             DISP=(NEW,PASS)
//CIGSNAP  DD  DSN=&&SNAP,
//             SPACE=(TRK,(1,1)),
//             DCB=(BLKSIZE=31200,LRECL=80,RECFM=FB,DSORG=PS),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//CIGLOG   DD  SYSOUT=*
//*
//STEP3C   EXEC PGM=CIGFOPCL
//* CIG AUTHORIZED LIBRARY
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGOUT   DD  DSN=&&TEMP2,
//             DCB=(BLKSIZE=13300,LRECL=133,RECFM=FBA),
//             SPACE=(CYL,(10,10)),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//ENDIF3   ENDIF
//* ------------------------------------------------------------------*
//*  STEP4A: ALLOCATE A TEMP FILE USED FOR FULL LOADS ONLY.           *
//* ------------------------------------------------------------------*
//STEP4AIF  IF   (STEP1.RC = 0) THEN
//STEP4A   EXEC  PGM=IEFBR14
//CIGPRINT DD  DSN=&&TEMP2,
//             DCB=(BLKSIZE=13300,LRECL=133,RECFM=FBA),
//             SPACE=(CYL,(10,10)),
//             UNIT=TDISK,
//             DISP=(NEW,PASS)
//SYSPRINT DD  SYSOUT=*
//ENDIF4A  ENDIF
//* ------------------------------------------------------------------*
//*  STEP4: CALL ENDEVOR AND PROCESS PRINT STATEMENTS.                *
//*         NOTE THIS STEP WILL BE EXECUTED FOR BOTH FULL AND         *
//*         INCREMENTAL LOADS.                                        *
//* ------------------------------------------------------------------*
//STEP4IF   IF   (STEP1.RC = 0 | STEP3.RC < 4) THEN
//* ENDEVOR AUTHLIB
//* CIG AUTHORIZED LIBRARY
//* ENDEVOR CONLIB
//STEP4    EXEC PGM=NDVRC1,PARM='C1BM3000',REGION=0M
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//        DD DSN=QUAL1.QUAL2.AUTHLIB,DISP=SHR
//CONLIB  DD DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//BSTIPT01 DD  DSN=&&TEMP1,DISP=(OLD,DELETE)
//CIGPRINT DD  DSN=&&TEMP2,DISP=(OLD,PASS)
//C1MSGS1  DD  SYSOUT=*
//ENDIF4   ENDIF
//* ------------------------------------------------------------------*
//* STEP 5A: PROCESS SNAP LABEL TAG COLLECTION. THESE WILL BE REAPPLIED
//*          AFTER THE FLOAD UPDATES HAVE OCCURRED.
//* USAGE:   NON-GREENHOUSE USERS CAN DELETE THIS STEP.
//* ------------------------------------------------------------------*
//STEP5IF  IF  (STEP4.RC < 8 | STEP3.RC < 5) THEN
//STEP5A   EXEC PGM=CIGTAG09
//* CIG AUTHORIZED LIBRARY
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
```

```
//CIGRPT   DD  SYSOUT=*
//CIGLOG   DD  SYSOUT=*
//CIGIN    DD  DSN=&&SNAP,DISP=(OLD,DELETE)
//CIGSNAP  DD  DSN=&&TAGSYN,
//             DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB),
//             SPACE=(CYL,(1,1)),
//             UNIT=TDISK,
//             DISP=(NEW,PASS,DELETE)
//* ------------------------------------------------------------------*
//* STEP 5:  PROCESS FLOAD, FDELETE, AND ENDEVOR PRINT STATEMENTS.
//*          (THE FASTLIST DATABASE WILL BE UPDATED HERE.)
//*          NOTE IF NOTHING QUALIFIED FOR THE INCREMENTAL LOAD
//*           THIS STEP WILL BE A PASS THROUGH FOR LOADS, BUT
//*           DELETES MAY STILL OCCUR.
//* ------------------------------------------------------------------*
//STEP5    EXEC PGM=CIGFLOD3
//* CIG AUTHORIZED LIBRARY
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGIN    DD  DSN=&&SYNTAX,DISP=(OLD,DELETE)
//CIGFDELE DD  DSN=&&TEMP1D,DISP=(OLD,DELETE)
//CIGINRPT DD  DSN=&&TEMP2,DISP=(OLD,DELETE)
//CIGLOG   DD  SYSOUT=*
//*CIGRPT   DD  SYSOUT=*  <== UNCOMMENT FOR DETAIL REPORT
//ENDIF5   ENDIF
//* ------------------------------------------------------------------*
//* STEP 6:  REAPPLY TAG ELEMENT SYNTAX.
//* USAGE:   NON-GREENHOUSE USERS CAN DELETE THIS STEP.
//* ------------------------------------------------------------------*
//STEP6IF  IF  (STEP5A.RC = 0) THEN
//STEP6    EXEC PGM=CIGTAG09
//* CIG AUTHORIZED LIBRARY
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGLOG   DD  SYSOUT=*
//CIGRPT   DD  SYSOUT=*
//CIGIN    DD  DSN=&&TAGSYN,DISP=(OLD,PASS)
//ENDIF6   ENDIF
//* ------------------------------------------------------------------*
//* STEPS 7 AND 8:  BUILD THE 'EE' TYPE CONTROL DATA RECORDS.
//* USAGE:   THIS STEP IS REQUIRED FOR CCID AND COMMENT VALIDATION.
//* NOTE1:   CIGDBJ15 IS THE STAND ALONE JCL FOR THESE TWO STEPS.
//* ------------------------------------------------------------------*
//* ------------------------------------------------------------------ *
//*                                                                    *
//* STEP7: PERFORM BATCH ADMIN FUNCTIONS AGAINST SELECTED ENVIRONNENTS*
//*                                                                    *
//* ------------------------------------------------------------------ *
//STEP7IF  IF  (STEP5.RC < 12) THEN
//STEP7    EXEC PGM=NDVRC1,PARM='ENBE1000',DYNAMNBR=1500,REGION=0M
//* ENDEVOR AUTHLIB
//* ENDEVOR CONLIB
//STEPLIB DD DSN=QUAL1.QUAL2.AUTHLIB,DISP=SHR
//CONLIB  DD DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//C1MSGS1   DD  SYSOUT=*
//C1MSGS2   DD  SYSOUT=*
//OUTFILE   DD  DSN=&&BATDATA,DISP=(NEW,PASS),
//          SPACE=(CYL,(10,10)),UNIT=TDISK,
//          DCB=(LRECL=80,BLKSIZE=31200,RECFM=FB)
//ENESCLIN  DD  *
 BUILD SCL FOR SYSTEM *
   FROM ENVIRONMENT 'YOUR-ENVIRONMENT-NAME'
   INCLUDE SUBORDINATES
   TO DDNAME 'OUTFILE'
 .
```

```
//* ------------------------------------------------------------- *
//*                                                               *
//* STEP8: PERFORM CONTROL DATA UPDATE TO FASTLIST DATABASE.      *
//*                                                               *
//* ------------------------------------------------------------- *
//STEP8IF  IF  (STEP7.RC < 12) THEN
//* CIG AUTHORIZED LIBRARY
//STEP8    EXEC PGM=CIGELOAD,REGION=0M
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGIN     DD  DSN=&&BATDATA,DISP=(OLD,DELETE)
//CIGLOG    DD  SYSOUT=*
//CIGPUNCH  DD  SYSOUT=*
//CIGDB     DD  DSN=FLHQ1.FLHQ2.FLSTDB,DISP=SHR
//STEP7END  ENDIF
//STEP8END  ENDIF
```

Deleting elements from the FastLIST database —FDELETE (CIGJCL10)

You can remove an element from the FastLIST database by using the FDELETE utility described below. When executing the FDELETE utility, all components associated with the element being deleted from the FastLIST database will also be deleted. Related element components include input components, output components, processor components, CCIDs, and related objects.

FDELETE is not constrained by the FILTERS included in your CIGINI file. FDELETE will delete any elements you specify in the FDELETE syntax from the FastLIST database.

FDELETE has no effect on the contents of Endevor.

Syntax

FDELETE ELEMENT element [*THROUGH element*]
FROM ENVIRONMENT environment
SYSTEM system
SUBSYSTEM subsystem
TYPE type
STAGENUMBER stage – number •

`FDELETE ELEMENT:`	Required. Identifies the element range to be deleted from the FastLIST database. Wild-carding is allowed.
`FROM:`	Required. The inventory location can be wild-carded using an "*". For example, specifying SYSTEM * will cause all systems to be deleted from the FastLIST database.
`Period:`	Required. The period terminates the FDELETE statement.

Model JCL

Member: CIGJCL10
Location: JCL Library

```
//**(JOBCARD)
//**
//* ---------------------------------------------------------------------*
//* NAME: CIGJCL10                                                       *
//*                                                                      *
//* PURPOSE: THIS JCL TO BE USED TO DELETE ELEMENTS FROM THE FASTLIST    *
//*          DATABASE AS PER THE FDELETE SYNTAX.                         *
//*                                                                      *
//* ---------------------------------------------------------------------*
//* ---------------------------------------------------------------------*
//*         * * *   N O T I C E   * * *                                  *
//* THIS JCL    IS A PROPRIETARY PRODUCT OF CHICAGO INTERFACE            *
//* GROUP, INC. @ COPYRIGHT 2005 CHICAGO INTERFACE GROUP, INC.           *
//* ALL RIGHTS RESERVED.                                                 *
//*                                                                      *
//* ---------------------------------------------------------------------*
//* REQUIRED JCL MODIFICATION:                                           *
//*   1) INCLUDE A JOBCARD                                               *
//*   2) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.         *
//*      - FLHQ1 AND FLHQ2                                               *
//*   3) MODIFY THE MEMBER FDELETE LOCATED IN SAMPLIB.                   *
//* ---------------------------------------------------------------------*
//* ---------------------------------------------------------------------*
//*  STEP1:  READ FDELETE STATEMENTS AND DELETE ELEMENTS FROM THE        *
//*          FASTLIST DATABASE.                                          *
//* ---------------------------------------------------------------------*
//STEP1   EXEC PGM=CIGFDELE
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGIN   DD DSN=FLHQ1.FLHQ2.SAMPLIB(FDELETE),DISP=SHR
//CIGLOG  DD SYSOUT=*
//CIGRPT  DD SYSOUT=*
```

Database backup, reorganization, and restore (CIGDBJ04)

The FastLIST database should be reorganized after every database load to remove the normal CI splits that occur in VSAM files. In addition, running the job CIGDBJ04 will remove logically deleted records from the FastLIST database.

You will need to adjust space allocations in the JCL below based on the size of your FastLIST database.

Backup, reorganization, and restore steps

Step 1:	Delete temporary files allocated during a prior execution of the CIGDBJ04 job stream.
Step 2:	Copy the data portion of the FastLIST database into a sequential file using the IDCAMS REPRO function.
Step 3:	Merge the A-type database records with the other FastLIST database records. Also eliminate all logically deleted database records.
Step 4:	Delete the old FastLIST database, allocate an new FastLIST, and reload the newly allocated FastLIST database with records created in Step 3 above.

Model JCL

Member: CIGDBJ04
Location: JCL Library

```
//*   (JOBCARD)
//*
//*********************************************************************
//* NAME:    CIGDBJ04

//* PURPOSE: BACKUP, REBUILD INDEXES, AND REORGANIZE THE DATABASE
//* ------------------------------------------------------------------ *
//* THIS IS THE SAMPLE JCL FOR BACKING UP AND REBUILDING THE FASTLIST *
//* DATABASE. THIS JOB SHOULD BE RUN ON A REGULARLY SCHEDULED BASIS.  *
//*                                                                    *
//* PRIOR TO USAGE THE USER MUST DO THE FOLLOWING:                     *
//*                                                                    *
//*   - ADD A VALID JOB CARD                                           *
//*   - CHANGE ALL REFERENCES TO 'FLHQ1' TO AN APPROPRIATE HIGH LEVEL *
//*           QUALIFIER FOR DATASETS.                                  *
//*   - CHANGE UNIT=TDISK TO A VALID UNIT NAME (TEMPORARY)             *
//*   - CHANGE UNIT=DUNIT TO A VALID UNIT NAME (PERMANENT)             *
//*   - CHANGE DVOLSER TO A VALID VOLSER FOR VSAM                      *
//*   - CHANGE SIZE PARAMETERS FOR EACH STEP TO MEET REQUIREMENTS      *
//*                                                                    *
//* ------------------------------------------------------------------ *
//* ------------------------------------------------------------------ *
//*                                                                    *
//* STEP1: DELETE OLD BACKUP AND OLD SORT FILE FROM PREVIOUS RUN.      *
//*                                                                    *
//* ------------------------------------------------------------------ *
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
     DELETE 'FLHQ1.FASTLIST.BACKUP' PURGE
//* ------------------------------------------------------------------ *
//*                                                                    *
//* STEP2: CREATE A SEQUENTIAL BACKUP OF THE FASTLIST DATABASE USING   *
//*        STANDARD IDCAMS REPRO SERVICES.                             *
//* NOTE >>> ONLY BACKING UP THE DATA PORTION                          *
//* ------------------------------------------------------------------ *
//STEP2    EXEC PGM=IDCAMS
//INDD02   DD DSN=FLHQ1.FASTLIST.DATABASE.DATA,DISP=OLD,
//            AMP='BUFNI=10,BUFND=10'
//OUTDD02  DD DSN=FLHQ1.FASTLIST.BACKUP,
//         DISP=(NEW,CATLG,DELETE),
//         UNIT=DUNIT,SPACE=(CYL,(10,5),RLSE),
//         DCB=(RECFM=VB,LRECL=2454,BLKSIZE=6160)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  REPRO IFILE(INDD02)  OFILE(OUTDD02)
/*
//* ------------------------------------------------------------------ *
//* STEP3: SORT DATA PORTION AND OMIT LOGICALLY DELETED RECORDS        *
//*                                                                    *
//*       -INPUT TO THIS STEP WILL BE A SEQUENTIAL BACKUP OF THE       *
//*        DATA PORTION OF THE FASTLIST DATABASE AND ALTERNATE RECS     *
//*                                                                    *
//*       -ALL RECORDS FLAGGED AS '--DEL' WILL BE DELETED AT THIS      *
//*        TIME. (LOGICAL DELETES)                                     *
//*                                                                    *
//*       -OUTPUT FROM THIS STEP SHOULD BE USED AS INPUT TO THE        *
//*        FASTLIST DATABASE DELETE,DEFINE, REPRO STEP.                *
//*                                                                    *
//* ------------------------------------------------------------------ *
//STEP3    EXEC PGM=SORT
//SORTIN   DD DSN=FLHQ1.FASTLIST.BACKUP,DISP=SHR
//SORTOUT  DD DSN=&&NEWFILE,DISP=(NEW,PASS,DELETE),
//         UNIT=TDISK,SPACE=(CYL,(10,5),RLSE),
//         DCB=(RECFM=VB,LRECL=2454,BLKSIZE=6160)
//SYSOUT   DD SYSOUT=*
//SORTMSG  DD SYSOUT=*
//SORTWK01 DD   UNIT=TDISK,SPACE=(CYL,(25,25))
//SORTWK02 DD   UNIT=TDISK,SPACE=(CYL,(25,25))
//SORTWK03 DD   UNIT=TDISK,SPACE=(CYL,(25,25))
```

```
//SORTWK04 DD   UNIT=TDISK,SPACE=(CYL,(25,25))
//SYSIN    DD *
  SORT FIELDS=(5,72,CH,A)
  OMIT COND=(77,2,CH,EQ,X'0000',&,79,3,CH,EQ,C'DEL')
  RECORD TYPE=V,LENGTH=(2454,,,77)
  SUM FIELDS=NONE
/*
//* ------------------------------------------------------------------ *
//*                                                                    *
//* STEP4: DELETE AND DEFINE THE CURRENT FASTLIST DATABASE, REPRO      *
//*        THE BACKUP AND REBUILD THE FILE ( WITHOUT THE ALTERNATE     *
//*        INDEX)                                                      *
//*                                                                    *
//* ------------------------------------------------------------------ *
//STEP4   EXEC PGM=IDCAMS,COND=(0,LT,STEP3)
//SYSPRINT DD SYSOUT=*
//INDD01   DD DSN=&&NEWFILE,DISP=(OLD,DELETE)
//SYSIN   DD *
 DELETE FLHQ1.FASTLIST.DATABASE
 DEFINE CLUSTER -
    (NAME('FLHQ1.FASTLIST.DATABASE') -
          SPEED UNIQUE FREESPACE(30 30) -
    TRACKS(90 60) -
    VOLUMES(DVOLSER) -
    SHR(4 3) -
    KEYS(72 0)  -
    RECORDSIZE(77 2450)) -
  DATA  (CISZ(16000)) -
  INDEX (CISZ(4096))
 REPRO INFILE(INDD01) OUTDATASET('FLHQ1.FASTLIST.DATABASE')
/*
//* ------------------------------------------------------------------ *
//* EXPAND THE INDEX TO 2 LEVELS                                       *
//* ------------------------------------------------------------------ *
//STEP5    EXEC PGM=CIGVSM2L,PARM='FLHQ1.FASTLIST.DATABASE'
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//*
```

The purpose of this utility is to provide the users with a way to capture and reload all elements that were processed in the FastLIST exit 2 but did not complete exit 3 processing. The reasons for the incomplete transactions vary, but the most common is a space ABEND in a processor or job cancellation.

The CIGFSYNC utility reads all sync records written to the primary database. During exit 2 processing, the pending action records are written to the primary database. For move and transfer, both the target and the source records are written. During exit 3 processing, these records are deleted to signal that the transaction was complete and successful. Since the sync records are written to the primary database, those customers with more than one database must ensure that users have write authority to the primary database.

Earlier versions of CIGFEXIT (the FastLIST collector module) had the pending logic shut off. The pending logic is now enabled in CIGFEXIT and is required for sync processing. An optional zap will be required to shut off the pending record update.

This utility will read sync records, write a copy of the record to the CIGRPT ddname, build FLOAD syntax into the CIGFLOAD ddname, then delete the sync records out of the database. To prohibit the deletion of the records, include the //NODELETE ddname.

USAGE NOTES

The intended usage of this utility is to feed FLOAD syntax into the FLOAD process. However, it can be used in separate processes or as a diagnostic tool.

The following is the JCL required for CIGFSYNC execution. All the ddnames shown are required except for the NODELETE ddname. This DD is optional and will prevent the sync utility from deleting the sync records from the database.

Model JCL
Member: CIGJSYNC

```
//**(JOBCARD)
//**
//*********************************************************************
//* NAME:    CIGJSYNC                                                 *
//* PURPOSE: GET ALL DB@SYNC RECORDS, BUILD FLOAD SYNTAX              *
//*          AND THEN DELETE THE DB@SYNC RECORDS.                     *
//* NOTES:   THIS UTILITY CLEANS OUT THE RECORDS AS IT PROCESSES      *
//*          THEN. THE DEFAULT IS TO DELETE THE SYNC RECORDS FOR EACH *
//*          ELEMENT.  TO OVERRIDE THIS DEFAULT, THE USERS MUST       *
//*          INCLUDE A //NODELETE DD DUMMY STATEMENT.                 *
//*********************************************************************
//*                                                                   *
//*  THIS IS THE JCL TO RUN THE FASTLIST DATABASE SYNC UTILITY.       *
//*  NOTE THE STEPLIB DD IS THE DATASET THAT CONTAINS THE TWO         *
//*  REQUIRED FASTLIST MODULES.                                       *
//*                                                                   *
//* REQUIRED JCL MODIFICATION:                                        *
//*   1) INCLUDE A JOBCARD                                            *
//*   2) CHANGE THE FOLLOWING AS PER THE INSTALLATION WORKSHEET.      *
//*      - FLHQ1 AND FLHQ2                                            *
//*                                                                   *
//*********************************************************************
//SYNC    EXEC PGM=CIGFSYNC
//STEPLIB DD DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGLOG  DD SYSOUT=*
//CIGRPT  DD SYSOUT=*
//CIGFLOAD DD *
//*NODELETE DD  DUMMY
```

Return Codes:

0	Sync records were found and processed.
4	No sync records found in database or sync records found and delete was bypassed.
12	Error occurred. Supporting messages will show the error. Typical error would be a missing, required dd statement or security error on the erase function.

Validate syntax in Duplicate Element Rules file (CIGJCL14)

Prior to enabling the FastLIST Duplicate Element Prevention Facility, you should verify that the Duplicate Element Rules syntax is correct. A detailed discussion of the Duplicate Element Prevention Facility is discussed earlier in chapter 5.

Invalid syntax in the Duplicate Element Rules file will prevent Endevor from executing actions.

Syntax

Refer to chapter 5 for a detailed discussed of the syntax defined to the Duplicate Element Rules file. Input to the program CIGFDUPL is defined via the CIGIN dd statement.

Model JCL
Member: CIGJCL14

```
//*   (JOBCARD)
//*
//*********************************************************************
//* NAME:    CIGJCL14
//* PURPOSE: VALIDATE SYNTAX IN THE DUPLICATE ELEMENT RULES FILE
//*********************************************************************
//* ------------------------------------------------------------------ *
//* ------------------------------------------------------------------*
//*         * * *   N O T I C E   * * *                                *
//* THIS JCL    IS A PROPRIETARY PRODUCT OF CHICAGO INTERFACE          *
//* GROUP, INC. @ COPYRIGHT 1998 CHICAGO INTERFACE GROUP, INC.         *
//* ALL RIGHTS RESERVED.                                               *
//*                                                                    *
//* ------------------------------------------------------------------*
//*  TO USE THIS JCL, YOU MUST:                                        *
//*         1) INSERT A VALID JOB CARD                                 *
//*         2) MAKE SURE THAT THE STEPLIB INCLUDES THE DATASET THAT    *
//*            CONTAINS CIGINI
//*         3) CHANGE FLHQ1, FLHQ2, QUAL1 AND QUAL2 AS PER YOUR        *
//*            INSTALLATION SHEET                                      *
//*         4) MAKE SURE THE CIGIN DATASET CONTAINS DUPLICATE
//*            ELEMENT RULES SYNTAX STATEMENTS
//* ------------------------------------------------------------------*
//STEP1    EXEC  PGM=CIGFDUPL
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGIN    DD  DSN=QUAL1.QUAL2.SYNTAX,DISP=SHR
//CIGLOG   DD  SYSOUT=
```

Extract and Load Endevor Inventory (CIGDBJ15)

FastLIST Release 12.0 supports inventory and processor group displays and CCID/Comment verification from the ISPF front end. To gain this functionality, the users must pre load Endevor inventory attributes. Please review the following JCL and modify as per the changes implemented in CIGDBJ15. Update all space and unit parameters prior to submitting the job.

The first step of this job invokes the Endevor Batch Administration utility to extract inventory definitions. The example syntax provided extracts all information for an environment. This extract can also be restricted to any level desired and can be run multiple times to achieve a complete load of information. The minimum extracts required for full functionality are System, Subsystem, Type, and Processor Group definitions.

Model JCL

MEMBER NAME CIGDBJ15
LOCATION JCL Library

```
//*
//* (JOBCARD)
//*
//* ------------------------------------------------------------------ *
//* NAME:      CIGDBJ15                                                *
//* PURPOSE:   THE PURPOSE OF THIS JCL IS TO POPULATE THE FASTLIST     *
//*            DATABASE WITH INVENTORY AND CONTROL DATA FROM ENDEVOR.  *
//* USAGE:     THIS UTILITY SHOULD BE RUN EVERY TIME THE ENDEVOR       *
//*            INVENTORY STRUCTURES AND MODIFIED.  SUGGEST RUNNING IT  *
//*            AS PART OF THE WEEKLY MAINTENANCE JOB FOR FASTLIST.     *
//* ------------------------------------------------------------------  *
//*                                                                     *
//* MODIFY THIS JCL TO MEET YOUR SITE REQUIREMENTS IN THE FOLLOWING     *
//* WAYS.                                                               *
//*                                                                     *
//*            1) INCLUDE A JOBCARD                                     *
//*            2) CHANGE FLHQ1 AND FLHQ2 AS PER WORKSHEET.              *
//*            3) CHANGE QUAL1 AND QUAL2 AS PER WORKSHEET.              *
//*            4) CHANGE TDISK TO A VALID DISK DEVICE UNIT NAME.        *
//*            5) CUSTOMIZE STEP2 TO INCLUDE YOUR FASTLIST DATABASE     *
//*               NAME.                                                 *
//*            6) CUSTOMIZE STEP1 INPUT TO INCLUDE EVERY ENVIRONMENT    *
//*               THAT IS TO BE EXTRACTED.  USE ONE BLOCK OF SYNTAX     *
//*               PER ENVIRONMENT.                                      *
//*               OPTIONALLY, THE BUILD SYNTAX CAN BE BROKEN UP INTO    *
//*               MULTIPLE JOBS.  BUILD SYNTAX FOR ALL SYSTEMS,         *
//*               SUBSYSTEMS, TYPES AND PROCESSOR GROUPS MUST BE        *
//*               GENERATED TO CREATE CONTROL RECORDS.  PROCESSOR       *
//*               SYMBOLICS ARE NOT REQUIRED.                           *
//* ------------------------------------------------------------------ *
//*                                                                     *
//* STEP1: PERFORM BATCH ADMIN FUNCTIONS AGAINST SELECTED ENVIRONENTS.  *
//* STEP2: PERFORM CONTROL DATA UPDATE TO FASTLIST DATABASE.            *
//*                                                                     *
//* ------------------------------------------------------------------ *
//* ------------------------------------------------------------------ *
//*                                                                     *
//* STEP1: PERFORM BATCH ADMIN FUNCTIONS AGAINST SELECTED ENVIRONNENTS  *
//*                                                                     *
//* ------------------------------------------------------------------ *
//STEP1    EXEC PGM=NDVRC1,PARM='ENBE1000',DYNAMNBR=1500
//* ENDEVOR AUTHLIB
//* ENDEVOR CONLIB
//STEPLIB   DD  DSN=QUAL1.QUAL2.AUTHLIB,DISP=SHR
```

```
//CONLIB    DD  DSN=QUAL1.QUAL2.CONLIB,DISP=SHR
//C1MSGS1   DD  SYSOUT=*
//C1MSGS2   DD  SYSOUT=*
//OUTFILE   DD  DSN=&&BATDATA,DISP=(,PASS,DELETE),
//          SPACE=(CYL,(10,10)),UNIT=TDISK,
//          DCB=(LRECL=80,BLKSIZE=31200,RECFM=FB)
//ENESCLIN  DD  *
 BUILD SCL FOR SYSTEM *
   FROM ENVIRONMENT 'ENV-NAME'
   INCLUDE SUBORDINATES
   TO DDNAME 'OUTFILE'
.
//* ------------------------------------------------------------------ *
//*                                                                    *
//* STEP2: PERFORM CONTROL DATA UPDATE TO FASTLIST DATABASE.           *
//*                                                                    *
//* ------------------------------------------------------------------ *
//STEP2IF  IF  (STEP1.RC < 16) THEN
//STEP2    EXEC PGM=CIGELOAD,REGION=0M
//* CIG AUTHORIZED LIBRARY
//STEPLIB   DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGIN     DD  DSN=&&BATDATA,DISP=SHR
//CIGLOG    DD  SYSOUT=*
//CIGPUNCH  DD  SYSOUT=*
//CIGDB     DD  DSN=FLHQ1.FLHQ2.FLSTDB,DISP=SHR
//STEP2END  ENDIF
```

CIGDBJ15 Return Codes

RC = 00	Request completed successfully.
RC = 12	For both steps, all serious error conditions. Missing file, etc. Check log for messages.

FastLIST 12.0

Chapter 7: Debugging Aids

Available Traces

Trace Facility

FastLIST includes a trace facility to provide additional detailed informational and data stream messages to assist in the debugging process. There are three traces available with FastLIST:

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

CIGTRACE

- Batch: //CIGTRACE DD DUMMY

TSO: ALLOCATE FI(CIGTRACE) DSN(*) SHR REUSE

This ddname causes general trace messages to be written to the CIGLOG dd statement. For example, allocation messages will appear in the CIGLOG file when CIGRACE is present in the file.

CIGOUT

- Batch: //CIGOUT DD SYSOUT=*

TSO: ALLOCATE FI(CIGOUT) DSN(*) SHR REUSE

This ddname will cause FastLIST Collector messages to be written to the CIGOUT file. Information written to this dd statement is useful in determining if the FastLIST Collector is active.

This dd statement can be coded when executing CA-Endevor.

CIGVTRAX

- Batch: //CIGVTRAX DD SYSOUT=*

TSO: ALLOCATE FI(CIGVTRAX) DSN(*dsname*) SHR REUSE

This ddname will turn on a VSAM database tracing facility. This facility is useful in verifying READ and WRITE functions being requested of VSAM when accessing the FastLIST or Package Utilities database. You must allocate CIGTRACE to activate CIGVTRAX.

CIGPTRAX

- Batch: //CIGPTRAX DD SYSOUT=*

TSO: ALLOCATE FI(CIGPTRAX) DSN(*dsname*) SHR REUSE

This trace shows the logic executed when tracing FastLIST syntax statements including the CIGINI module. Use this trace if a syntax error is flagged but the reason for the failure is not clear. This dd statement can be coded when executing FastLIST, Package Utilities, or CA-Endevor. You must allocate CIGTRACE to activate CIGVTRAX.

CIGFTRAX — This dd statement will show tracing of database activity against a FOR clause specified in the CIGINI module.

To enable, allocate a CIGFTRAX ddname to SYSOUT = * in your CLIST or JCL. You must allocate CIGTRACE to activate CIGVTRAX.

CIGZTRAX — This dd statement will show tracing of VSAM related activity.

To enable, allocate a CIGZTRAX ddname to SYSOUT = * in your CLIST or JCL. You must allocate CIGTRACE to activate CIGVTRAX.

Example:

The following JCL illustrates activating general tracing (CIGTRACE), VSAM tracing (CIGVTRAX), and parser syntax tracing (CIGPTRAX):

```
//FLIST  EXEC  PGM=CIGFLIST
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGLOG   DD  SYSOUT=*
//CIGIN    DD  *
       FLIST ELEMENT *
          FROM ENV TEST SYSTEM *   SUB *
               TYPE * STAGE NUM * .
//CIGTRACE DD  DUMMY
//CIGTRAX  DD  SYSOUT=*
//CIGPTRAX DD  SYSOUT=*
```

PRINTINI Utility

PRINTINI is a program designed to help the user or installer verify the active CIGINI in the path. To use this utility, code a small CLIST as follows. Copy this CLIST into a SYSPROC dataset or execute from the ISPF option 6 panel.

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

Enter 'TSO PRINTINI" . The output from the PRINTINI request will be returned to your screen as follows:

```
---------------
 |   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
 COLLECT COMPS?. Y
 COLLECT CCIDS?. Y
 FG EXEC?....... Y
 FILTERS........ NONE
```

You can also run the PRINTINI utility in batch using the PRINTINI JCL

```
//* JOBCARD
//STEP1  EXEC  PGM=PRINTINI
//STEPLIB  DD  DSN=FLHQ1.FLHQ2.LOADLIB,DISP=SHR
//CIGPRINT DD  SYSOUT=*
```

LAX Utility

LAX is a program designed to help format online allocations. To use this utility, code a small CLIST as follows:

```
        PROC 0 DDNAME(NONE)
        IF &DDNAME = NONE THEN GOTO SHOWALL
        CALL 'FLHQ1.FLHQ2.LOADLIB(LAX)' 'DDNAME'
        GOTO DONE
SHOWALL: +
        CALL 'FLHQ1.FLHQ2.LOADLIB(LAX)'
DONE: +
        END
```

This CLIST will print out ddnames currently allocated to your TSO session.

To set up the CLIST you will need to:

1) modify the CLIST specifying the correct load library name; and
2) copy the CLIST to your CLIST library (ISRCLIB).

To invoke the CLIST, type TSO LAX at the command line. To check if a specific ddname is allocated to your session type TSO LAX ddname

For example, `TSO LAX SYSLIB`

The ISRDDN command provides significantly more functionality that the LISTA or LAX command is supplied by IBM. To invoke this command type TSO ISRDDN in the command area.

Reserved ddnames and Considerations

Reserved ddnames

Throughout this manual there are 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.	CIGFTRAX	Tracing of FOR clause.
2.	CIGIN	Application input ddname.
3.	CIGINI	Initialization module override name.
4.	CIGLOG	Utility program log dataset.
5.	CIGOUT	Endevor exit output.
6.	CIGPTRAX	PARSER internals trace ddname.
7.	CIGRPT	Output ddname for all reports.
8.	CIGTRACE	Application trace ddname.
9.	CIGVTRAX	VSAM internals trace ddname.
10.	CIGZTRAX	VSAM low-level trace ddname.

Support for IBM Hilight and Recovery Functions

The following changes to the ISRECIG ISPPLIB member to enable both Hilight and Recover functions. Note that the Recover Function will only work while editing permanent datasets.

❶ Copy CIG panel ISRECIG to another name for backup purposes.
❷ Copy panel SYS1.SISPPENU(ISREFR04) into ISRECIG.

At this point the change is functional, but the title line will show a temporary dataset name if that's what's used. To fix that, make the following change:

1. Immediately after the)INIT line, change/add the following lines as shown:

```
.ZVARS = '( ZVMODET CIGTTL ZCL ZCR ZCMD ZSCED)'

  &CIGTEMP = TRUNC(&ZTITLE,3)
  IF (&CIGTEMP = 'SYS')  /* TEMPORARY DATASET NAME */
    &CIGTTL                                                    =
'&VEDTEL/&VEDTEMV/&VEDTSYS/&VEDTSBS/&VEDTTYP/&VEDTSTG +
                          (&VEDTVV..&VEDTLL)'
  ELSE
    &CIGTTL = &ZTITLE
```

Table of Figures