Chapter
1

ENDEVOR CONCEPTS AND FACILITIES

SYS-ED/
Computer
Education
Techniques, Inc.

Objectives

You will learn:

- How Endevor builds and maintains software inventory components.
- How Endevor provides a logical structure for classifying software inventory.
- Endevor life cycle.
- Endevor libraries and functions.

1 Application Development Life Cycle: Control, Automation, and Monitoring

Endevor is a software package that controls, automates, and monitors the application development life cycle. It provides the capability for automating and controlling the movement of source code from development through to production.

Endevor is used for:

- Automatically comparing and tracking changes made by a programmer to the production version, creating an on-line change history.
- Preventing conflicting changes to the same system component.
- Browsing and manipulating all components relating to an application from a single screen, saving time and ensuring that changes are complete.
- Automating creation of executables.
- Ensuring that the source, executable, and any other form of an element and module correspond.
- Applying consistent procedures, including the automation of compiles, impact analysis, and standards-checking functions, to any component type.
- Processing changes to packages and approvals on-line, eliminating change-related paperwork.
- Viewing or retrieving a prior level of any element.
- Reporting on element definition, content, and change history.
- Enforcing change control procedures.

2 Endevor: Inventory of Software Components

Endevor maintains an inventory of software components - JCL, source, procs, copy books, etc. - for all applications.

With Endevor:

- Each application will have the ability to determine the procedure for promotion of all of its component elements.
- The source module and the load module will always match.
- There is the capability for program reversion back to a specific change level.
- Each program will be compiled the same way every time.
- Users can be involved in the approval process.

Endevor accelerates the debugging process and provides a framework for viewing changes that have been made: who, what, when, where, and how.

3 How Endevor Works

Endevor uses a base/delta technology format.

The base is the original production version of the element as it was first loaded into the Endevor system, without any modifications.

The deltas contain only the changes that have been made to the base level code. There is one delta version for every change level.

When a specific version of an element or module is displayed or retrieved, Endevor first makes a copy of the base version. It then applies all updates from the subsequent deltas to build the version level that has been selected.

Members are maintained by version and level numbers.

Endevor has a logical inventory scheme for all elements. Each element is classified by system, subsystem, type, and element name.

Endevor will protect all source elements within its environment. There can be no updates to any production elements except through Endevor. Update access to any production load library is not allowed except through Endevor.

3.1 Endevor: Basic Operations

Normal change procedures made through Endevor include:

- Retrieving elements from production to a development library.
- Making changes to elements.
- Adding and updating elements in the test stage.
- Moving elements to QA.
- Moving elements back into production.

4 Inventory Structure

The CA-Endevor inventory structure provides:

- The ability to work with program modules without having to know where they are physically located or how they are compiled.
- A list of all the program components that make up an application, regardless of type.
- Ascertaining the location(s) of an element by entering the element name on a display screen.
- Acting on a cross section of a program inventory.

4.1 Building the Inventory Structure

The CA-Endevor administrator builds an inventory structure based on the stages in a site's software life cycle.

There are six steps in setting up an inventory structure:

1. Determine the stages in the software life cycle.
2. Decide which stages should be put under the control of CA-Endevor.
3. Define two-stage environments based on the decisions in Steps 1 and 2, and link these environments/stages together to form the map.
4. Define applications and systems for each stage.
5. Define specific applications and subsystems within each system.
6. Define the types present in each stage and processors for each.

5 Logical Structure

Endevor helps to manage the software life cycle by providing a consistent and flexible logical structure for classifying the software inventory.

There are six components to this inventory structure: environments, stages, systems, subsystems, types and elements.

Environment	Environments consist of functional areas within the life-cycle. Endevor can be set up with multiple environments. For example, Development, Beta or QA, and Production.
Stage	There are two stages in every environment. Stages have names representing their place in the life cycle. The stages are linked together in a logical structure that incorporates a specific promotion route for all elements within and between environments. These routes comprise the map for an Endevor system.
System	The system refers to related applications stored within Endevor. For example, Endevor can contain a financial system and a payroll system.
Subsystem	The subsystem is a specific application within a system. For example, the finance system may contain such subsystems as accounts payable, accounts receivable, and general ledger.
Type	The type of module refers to the category of the source code contained in the member or element. Some examples of types are: VS COBOL, COBOL II, Assembler, PROCs, JCL, and copy books. Specific processors are associated with the different types of source code where necessary.
Element	The element is another name for a member or module. Each element is classified by system, subsystem and type. Its environment and stage determine its location in the software life cycle.

An element is promoted from one stage to the next stage in the sequence.

The promotion routes established for software inventory at a site. Environments and stages are mapped to each other within an Endevor table. Systems, subsystems, types, and processor groups are mapped to each other on their respective definition panels.

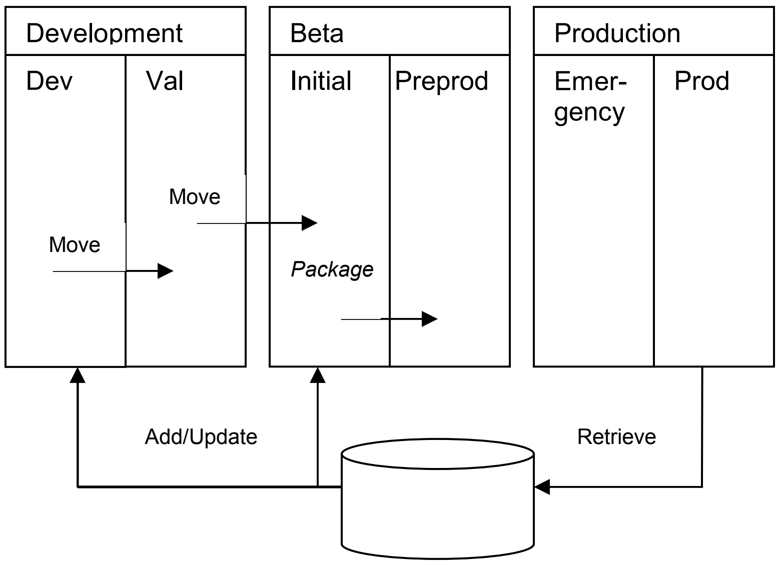
6 Processors

Processors are standard z/OS JCL job streams that manipulate elements and their outputs.

The three types of processors are:

GENERATE (compile)	Translates source code into executable modules and then moves the load module to the proper output libraries.
DELETE	Deletes outputs created by generate processors whenever an element is deleted, transferred, moved, or archived.
MOVE	Copies or regenerates output, element information, and component lists from the source location to the target location of a Move action.

7 Endevor Life Cycle



8 Package Processing

Sign-in is the assignment of a user-id to an element, establishing “ownership” of that element during the development process.

Sign-out is automatic when retrieving elements from or adding and updating elements into Endevor.

A package is a group of Endevor elements that requires approval before it can be executed.

Creating packages provides the capability to:

- Group specific elements so that they can be maintained and tracked as a single unit.
- Establish formal approval procedures to ensure data integrity throughout the modification process.
- Centralize specific element groups so that they may be seen across environments and may be reused at another level.

Casting a package permanently places the elements to be included in a package. The elements in a package cannot be edited or modified after it has been cast.

9 Endevor Libraries

In order to implement an inventory structure, the CA-Endevor administrator must define and allocate the following libraries.

Library	**Explanation**
Master Control File libraries	There is one MCF: Master Control File for every stage. A Master Control File stores system, subsystem, and type definitions, the names of the elements currently in that stage, and other information.
Package dataset	There is one package dataset per site. Endevor stores all packages built at the site and related information in this dataset.
Base and delta libraries	Endevor uses base and delta libraries to store source code. Base libraries store source code when it is first added to Endevor. Delta libraries store changes made to the source. There is one base and delta library associated with each type, but different types can share the same base and delta library.
Output libraries	Endevor uses output libraries to store executable forms of programs produced by processors.
Source output libraries	Endevor uses source output libraries to store copybooks, assembler macros, or JCL procedures that are copied elsewhere and therefore have to be available in full source form.
Processor load and listing libraries	Endevor uses processor load libraries to store the executable form of CA-Endevor processors. One processor load library is allocated for Stage 1 to use for testing, and a second processor for Stage 2 use in production.
Processor output libraries	These are libraries that are referred to in processors, to which processors write their output. Processor output libraries can be source libraries, executable libraries, or listing libraries.

10 Endevor Functions

Function	Explanation
ADD	Puts an external dataset member under CA-Endevor's control.
ARCHIVE	Writes the current version of an element to a sequential dataset.
COPY	Copies an element from an archive dataset to a dataset external to Endevor.
DELETE	Erases base and delta forms of an element and removes related information from a Master Control File.
DISPLAY	Displays information about an element.
LIST	Creates a list of elements that meet specific selection criteria.
MOVE	Moves elements between stages, within or across environments.
PRINT	Prints element or member information.
RESTORE	Restores elements to CA-Endevor from an archive dataset.
RETRIEVE	Copies elements from CA-Endevor to an external dataset.
SIGNIN	Removes the user signout associated with an element.
TRANSFER	Moves elements between locations that are not on the same map route.
UPDATE	Updates an element from an external dataset.