opentext™

ChangeMan ZMF

ERO Concepts

8.3

Table of Contents

1. About this Guide

ChangeMan® ZMF is a comprehensive and fully integrated solution for Software Change Management systems in z/OS environments. It provides reliable and streamlined implementation of software changes from development into production.

ChangeMan ZMF manages and automates the application life cycle, protects the integrity of the code migration process, and results in higher quality delivered code to any test environment and to the production environment.

Before You Begin. See the Readme for the latest updates and corrections for this manual.

Objective. ChangeMan ZMF ERO Concepts reviews release management concepts, defines Enterprise Release Option (ERO) terminology, traces a release life cycle in ERO, and introduces the ERO administrator's roles and responsibilities.

Audience. Use this document if you are responsible for any of these tasks:

- Managing software releases at your enterprise
- Managing test environments
- Developing and changing software components managed by ChangeMan ZMF

This guide assumes that you are familiar with ChangeMan ZMF and your security system.

Navigating this book. This manual is organized as follows:

Chapter 1 - Contains information about this manual.

Chapter 2 - Release Management

Chapter 3 - ERO Concepts

Chapter 4 - Roles and Responsibilities

Guide to ChangeMan ZMF Documentation

The following sections provide basic information about ChangeMan ZMF documentation.

ChangeMan ZMF Documentation Suite

The ChangeMan ZMF documentation set includes the following manuals.

Manual	Description
Customization Guide	Provides information about ChangeMan ZMF skeletons, exits, and utility programs that will help you to customize the base product to fit your needs.
Db2 Option Getting Started Guide	Describes how to install and use the Db2 Option of ChangeMan ZMF to manage changes to Db2 components.
ERO Concepts	Discusses the concepts of the ERO Option of ChangeMan ZMF for managing releases containing change packages.
ERO Getting Started Guide	Explains how to install and use the ERO Option of ChangeMan ZMF to manage releases containing change packages.
IMS Option Getting Started Guide	Provides instructions for implementing and using the IMS Option of ChangeMan ZMF to manage changes to IMS components.
INFO Option Getting Started Guide	Describes two methods by which ChangeMan ZMF can communicate with other applications: - Through a VSAM interface file. - Through the Tivoli Information Management for z/OS product from IBM.
Installation Guide	Provides step-by-step instructions for initial installation of ChangeMan ZMF. Assumes that no prior version is installed or that the installation will overlay the existing version.
Java / zFS Getting Started Guide	Provides information about using ZMF to manage application components stored in USS file systems, especially Java application components.
Load Balancing Option Getting Started Guide	Explains how to install and use the Load Balancing Option of ChangeMan ZMF to connect to a ZMF instance from another CPU or MVS image.
M+R Getting Started Guide	Explains how to install and use the M+R Option of ChangeMan ZMF to consolidate multiple versions of source code and other text components.
M+R Quick Reference	Provides a summary of M+R Option commands in a handy pamphlet format.
Messages	Explains messages issued by ChangeMan ZMF, SERNET, and System Software Manager (SSM) used for the Staging Versions feature of ZMF.
Migration Guide	Provides guidance for upgrading ChangeMan ZMF
OFM Getting Started Guide	Explains how to install and use the Online Forms Manager (OFM) option of ChangeMan ZMF.

Manual	Description
SER10TY User's Guide	Gives instructions for applying licenses to enable ChangeMan ZMF and its selectable options.
User's Guide	Describes how to use ChangeMan ZMF features and functions to manage changes to application components.

Using the Manuals

This section highlights some of the main Reader features. For more detailed information, see the Adobe Reader online help system. The PDF manuals include the following features:

- **Bookmarks.** All of the manuals contain predefined bookmarks that make it easy for you to quickly jump to a specific topic. By default, the bookmarks appear to the left of each online manual.
- **Links.** Cross-reference links within a manual enable you to jump to other sections within the manual with a single mouse click. These links appear in blue.
- **Comments.** All PDF documentation files that Serena delivers with ChangeMan ZMF have enabled commenting with Adobe Reader. Adobe Reader version 7 and higher has commenting features that enable you to post comments to and modify the contents of PDF documents. You access these features through the Comments item on the menu bar of the Adobe Reader.
- **Printing.** While viewing a manual, you can print the current page, a range of pages, or the entire manual.
- **Advanced search.** Starting with version 6, Adobe Reader includes an advanced search feature that enables you to search across multiple PDF files in a specified directory.

Searching the ChangeMan ZMF Documentation Suite

There is no cross-book index for the ChangeMan ZMF documentation suite. You can use the Advanced Search facility in Adobe Acrobat Reader to search the entire ZMF book set for information that you want. The following steps require Adobe Reader 6 or higher.

1. Download the ZMF All Documents Bundle ZIP file and the ZMF Readme to your workstation from the My Downloads tab on the Serena Support website.
2. Unzip the PDF files in the ZMF All Documents Bundle into an empty folder. Add the ZMF Readme to the folder.
3. In Adobe Reader, select Edit | Advanced Search (or press Shift+Ctrl+F).
4. Select the All PDF Documents in option and use Browse for Location in the drop-down menu to select the folder containing the ZMF documentation suite.
5. In the text box, enter the word or phrase that you want to find.

6. Optionally, select one or more of the additional search options, such as Whole words only and Case-Sensitive.
7. Click Search.
8. In the Results, expand a listed document to see all occurrences of the search argument in that PDF.
9. Click on any listed occurrence to open the PDF document to the found word or phrase.

Online Help

Online help is the primary source of information about ChangeMan ZMF. Online help is available as a tutorial, through Help screens, and in ISPF error messages.

Online Tutorial

ChangeMan ZMF includes an online tutorial that provides information about features and operations, from high-level descriptions of concepts to detailed descriptions of screen fields.

To view the tutorial table of contents, select option T from the Primary Option Menu, or jump to it from anywhere in ChangeMan ZMF by typing =T and pressing ENTER.

Press PF1 from anywhere in the Tutorial for a complete list of Tutorial navigation commands and PF keys.

Online Help Screens

If you have questions about how a ChangeMan ZMF screen works, you can view a help panel by pressing PF1 from anywhere on the screen.

Online Error Messages

If you make an invalid entry on a ChangeMan ZMF screen, or if you make an invalid request for a function, a short error message is displayed in the upper right corner of the screen. Press PF1 to display a longer error message that provides details about the error condition.

Remember that the long message does not display automatically. Request the long message by pressing PF1.

Typographical Conventions

The following typographical conventions are used in the online manuals and online help.

These typographical conventions are used to assist you when using the documentation; they are not meant to contradict or change any standard use of typographical conventions in the various product components or the host operating system.

Convention	Explanation
italics	Introduces new terms that you may not be familiar with and occasionally indicates emphasis.
**bold	Emphasizes important information and field names.
UPPERCASE	Indicates keys or key combinations that you can use. For example, press the ENTER key.
monospace	Indicates syntax examples, values that you specify, or results that you receive.
monospaced italics	Indicates names that are placeholders for values you specify; for example, filename.
vertical rule |	

2. Release Management

This chapter presents an overview of release management concepts.

Goals

The goal of release management is to provide you a way to organize and manage the release of your software applications; you establish a repeatable process to move applications to production so they are available to users.

These applications can range from simple applications to large complex systems with dependencies between components within applications.

Requirements

For effective release management, you must:

- Identify applications and their associated release dates, noting applications releasing concurrently.
- Understand the key relationships between application components. This becomes more difficult as the number of components increases.
- Accurately manage the dependencies among the applications and various releases of the applications.
- Ensure the proper components are available for any given release of an application.
- Provide environments for thorough testing by appropriate personnel.
- Ensure no collisions occur among applications during the testing and release schedules and during deployment, especially with applications releasing concurrently.
- Integrate and manage application changes into a larger system. For example, Accounts Payable and Accounts Receivable applications need to be migrated into the Finance system.
- Schedule release dates, identify resource constraints, and obtain management approval for upcoming releases.

Considerations

Release management affects personnel across your organization:

- Managers want to minimize the risk involved with placing new releases into production while maintaining timely release schedules. They need to consider software quality, time, and cost when determining what applications will be released, and when. For example, if software quality becomes critical, then release schedules and costs need to be adjusted.
- Developers need to accurately document complex and changing dependencies among their applications and their various releases. They need to evaluate these relationships to determine the impact of any changes to components.
- Users are involved in the installation and appropriate configuration of the application to their particular environment. They want applications that are easy to install, test, and deploy quickly. Quality, usability, and robust features are important when the application is available to end users.

Objectives

To mitigate risk and ensure properly tested applications are placed into production, the objectives of effective release management are to:

- Test applications together in a consolidated environment.
- Manage the dependencies among the applications and releases.
- Support multiple changes to the same component in various releases of an application.
- Resolve obstacles prior to production.

Strategies

Release management can be implemented in a physical manner. You establish separate sets of libraries at the application level for the development and testing of components.

These libraries of changed application components are then funneled, or consolidated, until only a single set of libraries exist with all changed components. The final deployment to production takes place from this single set of libraries.

The following topics consider each objective listed above.

Test Applications Together in a Consolidated Environment

Developers change components in development libraries belonging to individual developers or teams. The components are then moved to and tested in libraries belonging to their applications. With multiple changes to multiple applications occurring simultaneously, there is a good chance that the same components are changed in multiple places.

Concurrent development of these components is detected when you consolidate libraries, as applications are tested along their designated release migration path. The different versions of the same component have to be resolved to prevent overlays of the changes or loss of new function as these components move forward in the migration path.

For example, in the following diagram, Accounts Payable and Accounts Receivable applications are consolidated and migrated into a set of Finance libraries. When you consolidate libraries at the Finance level, you have to resolve different versions of the same component. From Finance, you migrate components to Integration, then to Final. You deploy to production from these Final libraries.

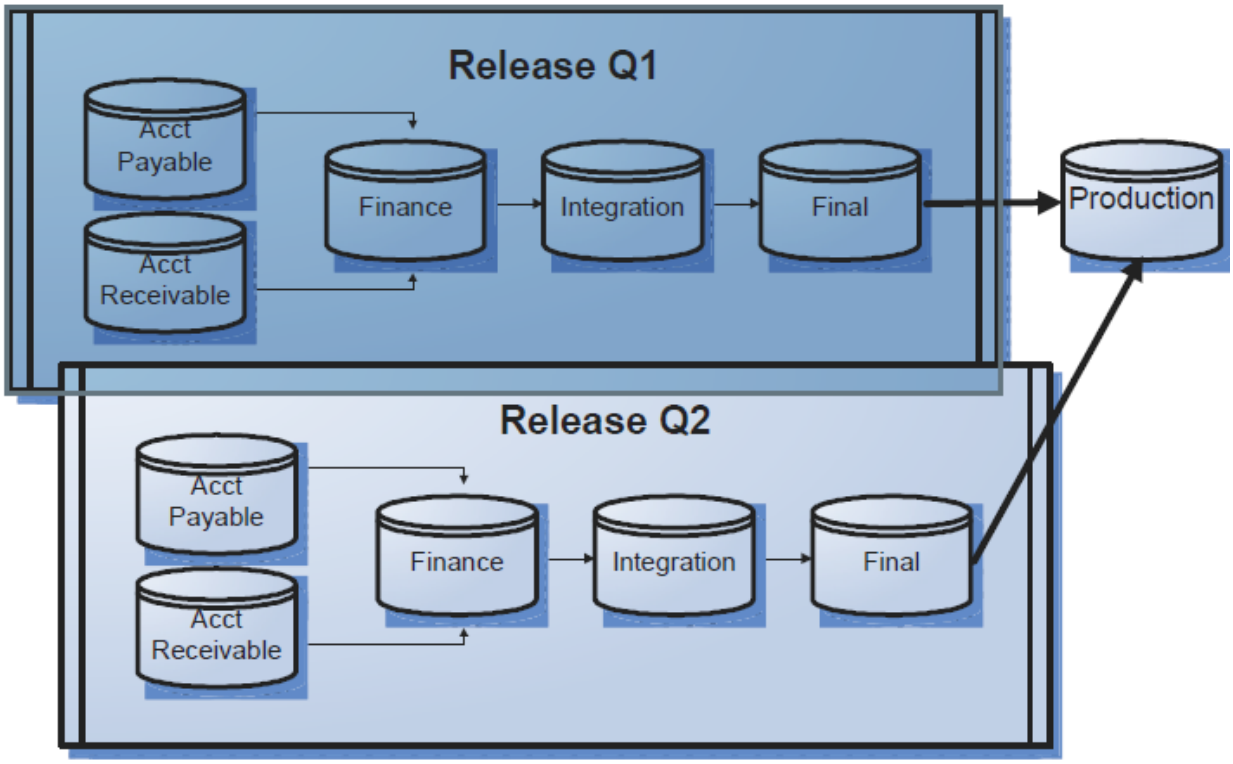

Figure 2-1. Release management includes consolidating libraries.

Manage the Dependencies among the Applications and Releases

Developers need to document the dependencies among applications, and evaluate the relationships between application components and other components they include. With multiple releases, there may be different versions of the same component with different dependencies. It can be a tedious process to document these relationships.

You can use utilities to help automate the documentation of component relationships. These utilities examine the relationships between source programs and copybooks and the relationships of load modules to the composite load modules that call them.

With a physical implementation, you can use these utilities to examine the components in libraries for a particular application, as well as libraries up and down the established migration path to determine relationships. You can also look across releases to evaluate the relationships between releases or in the contents of a release.

Resulting component relationship information can be stored in a repository. You can use this information for analysis purposes, and to detect and report issues in related applications and releases.

Support Multiple Changes to the Same Component in Various Releases of an Application

Components may be changed in each release, and these releases may be in motion simultaneously. Fixes for component defects in one release may need to be applied to all corresponding components in other releases.

With a physical implementation of release management, parallel development takes place in the application libraries for each release. These changes are isolated from any other application's or release's changes.

Concurrent development of these components is detected when you consolidate libraries as applications are tested along their designated release migration path. The different versions of the same component have to be resolved to prevent overlays of the changes or loss of new function as these components move up the migration path.

Resolve Obstacles prior to Production

As you are testing applications together in a consolidated environment, you may encounter application discrepancies. Components may produce unpredictable results. You may encounter operational issues with hardware and operating system incompatibilities. You need to remove these obstacles before components can be properly tested, approved, and moved forward in the migration path.

You can identify checkpoints along the way as consolidation occurs. When obstacles are encountered, you can block forward movement with applicable rules in place. You can establish rules to allow only successfully tested components to be moved forward. You can also require appropriate approvals from development, management, and users to ensure that correct changes are implemented at the appropriate times.

3. ERO Concepts

With the ChangeMan ZMF Enterprise Release Option (ERO), an extension to ChangeMan ZMF, you manage changes as a release. A release represents a collection of ChangeMan ZMF simple change packages from any number of applications that must be installed at the same time.

Introduction

Releases allow you to manage high volumes of changes, as hundreds of simple change packages can be attached to any given release. ERO maintains relationships between components and the packages that comprise a release. ERO manages multiple releases in motion simultaneously.

ERO extends the capability of ChangeMan ZMF in the following ways:

Support for release management, with checkpoints along the way

With the flexibility of ERO architecture, you define releases with life cycles that correspond to your release migration paths. The ERO release life cycle is controlled by a set of rules to help manage and protect your production environment. ERO offers:

- notifications at every major point in a release life cycle
- built-in approval processes
- query capabilities
- library concatenation definitions for build processes

Consolidation of migration paths to an integrated environment

ERO provides specific development paths for releases that progressively consolidate package components into a single set of libraries. Concurrent development of these package components is detected when you consolidate libraries. The different versions of the same component have to be resolved to prevent overlays of the changes or loss of new function as these components move up the migration path.

Release audit capabilities, with automatic resolution of out-of-sync conditions

Similar to package audit in the base ChangeMan ZMF product, release audit evaluates relationships between different versions of the same component. It also evaluates relationships between components and other components they include, like copybooks and statically linked load modules. However, release audit is more comprehensive than package audit. Release audit examines:

- components in libraries for a particular release

- components in libraries in prior releases that will be installed sooner
- components in baseline libraries

Automatic determination of copybook and load module concatenations for build processes

ERO gives you the flexibility to define copybook and load module concatenations for use in the build processes. These concatenations are pointed to by SYSLIB JCL statements. ERO manages these SYSLIB concatenations, even as library concatenations and contents in build jobs change when the schedule relationship between releases is changed. ERO also manages the SYSLIB concatenations when applications and packages are added to or removed from a release.

More information

Key Terms and Concepts

Advantages of ERO

Key Terms and Concepts

ERO (Enterprise Release Option) introduces new terms and concepts to ChangeMan ZMF.

Release

A release represents a collection of ChangeMan ZMF simple change packages from any number of applications that must be installed at the same time. In ERO, a release is a logical set of rules for relating the physical elements of a release. These elements are:

- the baseline libraries of the applications
- the release libraries
- the staging libraries of the change packages

Releases are managed by a Release Manager, who has exclusive authority over the release creation and life cycle. Release managers can get real-time information about components within releases online.

When you create releases, you define release install date and time ranges. All packages you attach to a release must have install dates within the release install range.

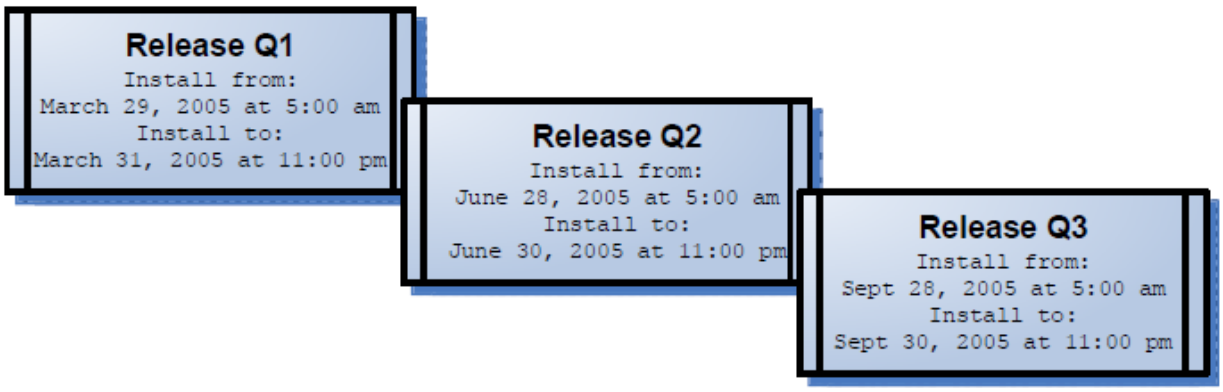

Figure 3-2. Releases are created with installation date and time ranges.

When you create a release, you define prior releases. This information can be updated as additional releases are added.

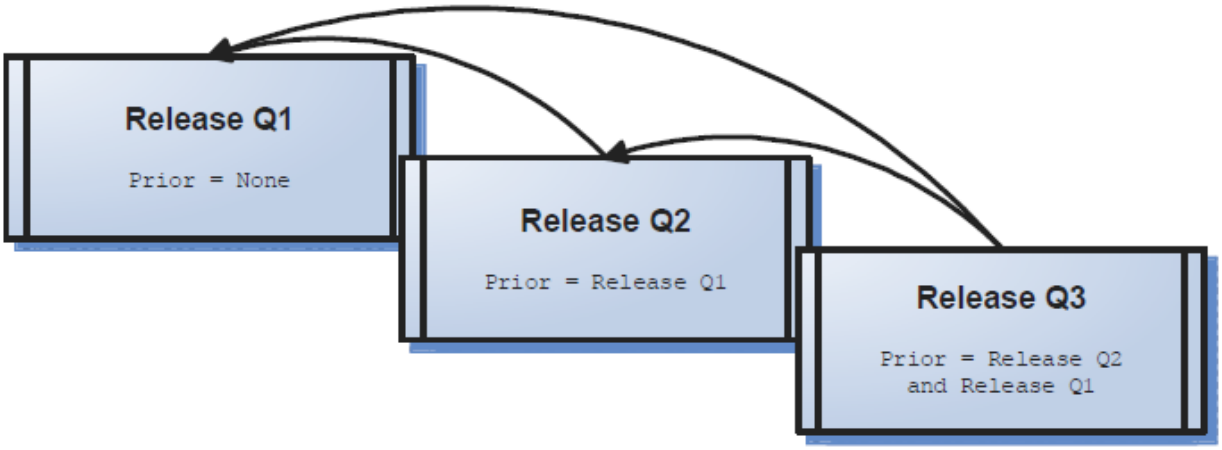

Figure 3-3. You define prior releases when you create a release.

Release Area

A release area is a set of component libraries that represents a migration step in the release life cycle. Each area has its own set of dynamically allocated libraries based on application and library type. The default data set name format for release area libraries is:

```
HLQ.ReleaseID.AreaID.ApplID.Libtype
```

The consolidation of components takes place in these release area libraries, which are used in SYSLIB concatenations for build processes in release packages. They are also used by release audit to validate release component relationships.

A minimum of two release areas are required. A starting area is used to check package components into a release. The final area is used to move the release into production, and it must contain the entire release. For a typical release, you define one or more starting areas, multiple intermediate areas, and a final area.

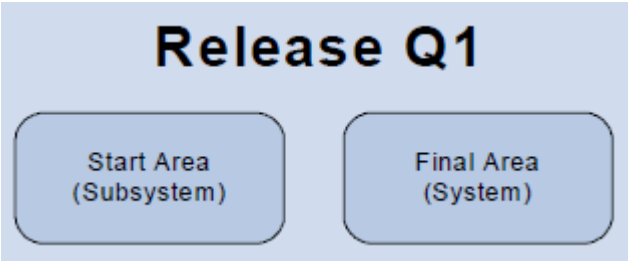

Figure 3-4. Releases require a minimum of two areas.

Starting areas are subsystem areas. Final areas are system areas. Intermediate areas can be either subsystem areas, which can be subordinate to subsystem and system areas, or system areas, which can be subordinate only to other system areas.

Area definitions include fields for prior and next areas, and approvals may be required for migration from one area to another. If you specify check-in approvals, they must be granted before components can be checked in to an area. If you specify check-off approvals, they are required before components can be migrated into the next area.

As components are migrated from one area to another, prior area libraries remain fully intact. This differs from base ChangeMan ZMF package promote, where package components are removed from libraries in the prior level.

Release Migration Path

The release migration path is defined by the prior and next areas specified in each release area definition.

Release area definitions include fields for prior and next areas. Starting areas, which serve as the entry point for packages, have only next areas, while final areas serve as the production migration point and have only prior area definitions. Additional areas can have both prior and next areas. You define as many areas as you need to establish your release migration path.

In the following diagram, Start Area has no prior area, but has one next area (Area2). Area2 is defined with a prior area (Start Area) and a next area (Area3). Likewise, Area3 is defined with a prior area (Area2) and a next area (Final Area). Final Area is defined with one prior area (Area3), and has no next area.

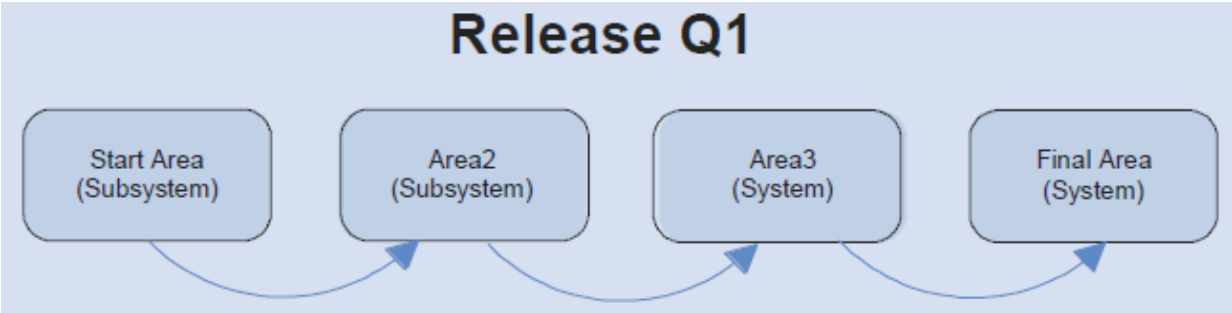

Figure 3-5. Release migration paths are defined when you link areas.

You can define multiple migration paths within a release. Multiple starting areas migrate to other areas, gradually combining until they come together in a single final area. Installation copies components from the final area into production and updates the baseline libraries of the applications in the release. The baseline update process ripples current and prior versions of package components down in the stack of prior baseline versions. The package components are then copied into the baseline libraries as the new current version.

In the following diagram, Team1 and Team2 changes are migrated and consolidated to the Group1 area, and Team3 and Team4 changes to the Group2 area. Both Groups are migrated and consolidated to the Integration area before migration to the Final Area. The Fixes Area is used as a quick path to the Integration and Final areas.

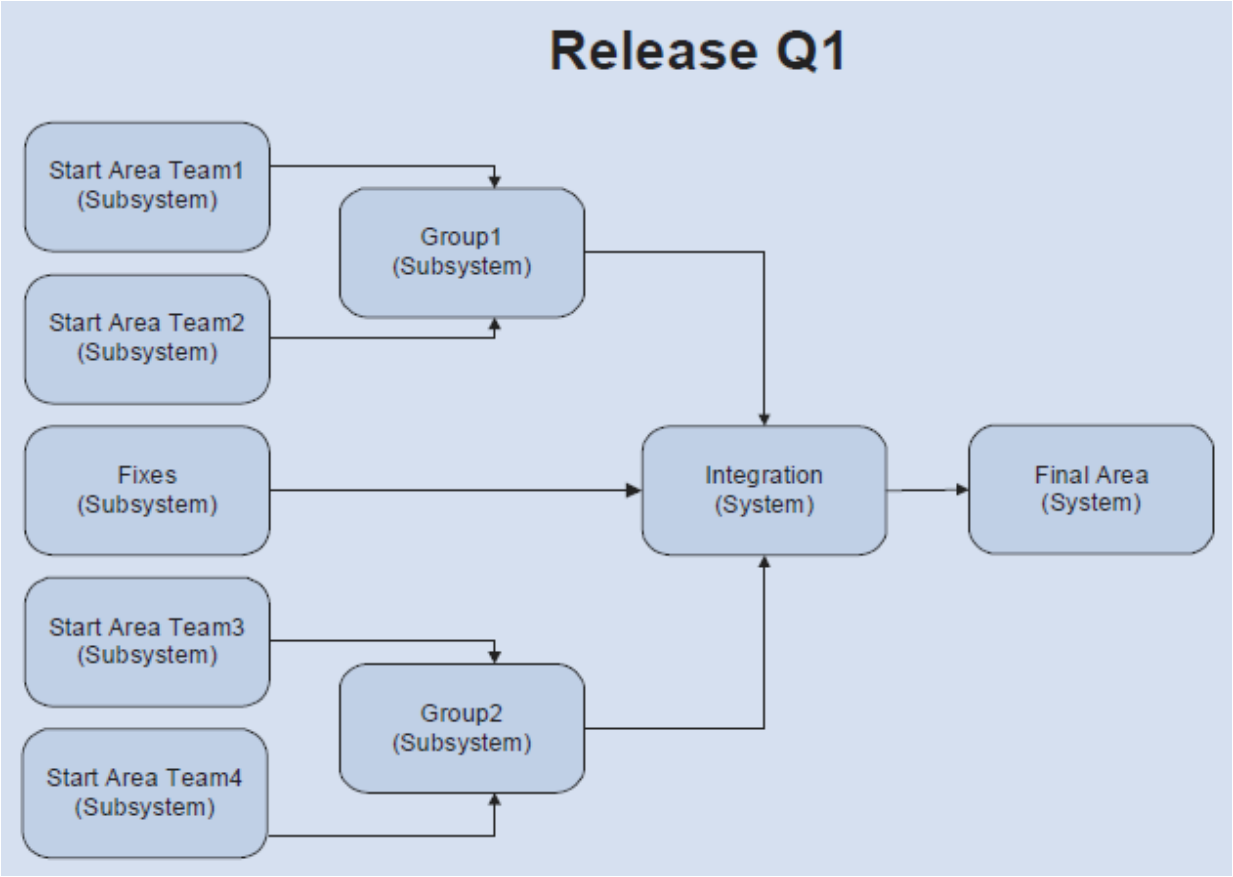

Figure 3-6. Multiple migration paths can be defined in a release.

As shown in the diagram above, you can set up migration paths for change requests, which follow a longer and more robust migration path, as well as fixes, which require a shorter process to production.

Release Application

An application becomes a release application when it is joined to a release.

Applications are joined to a release by an applications administrator, who also chooses what library types from each joined application are included in the release. By restricting library types in a release, administrators can build special purpose releases. For example, an online release would contain only the library types associated with CICS components. A batch release would contain library types associated with JCL and batch processing.

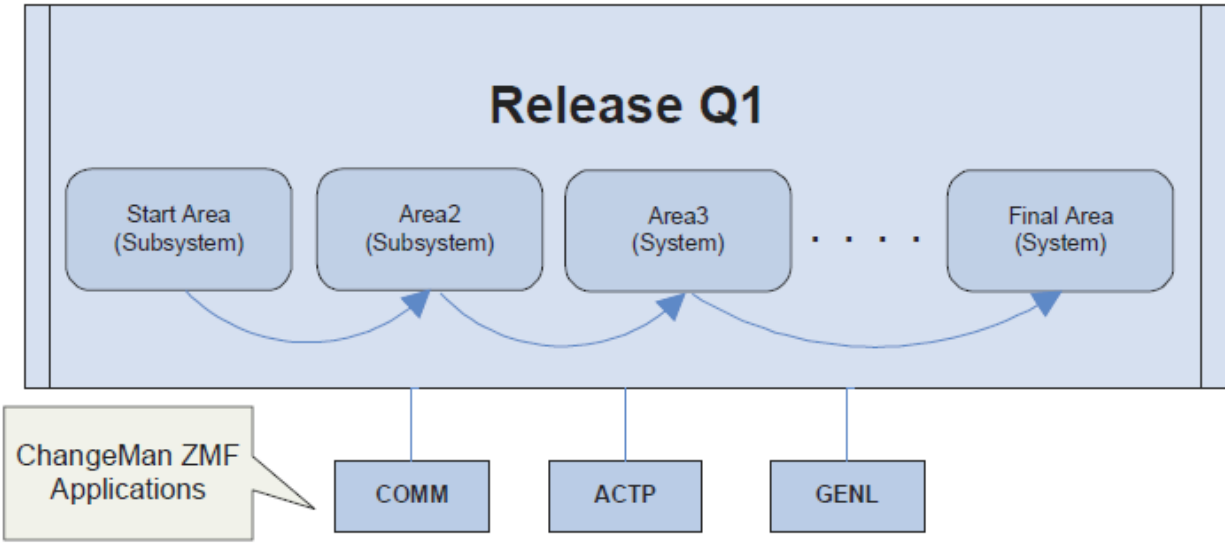

Figure 3-7. Applications are joined to a release.

By joining applications to a release, the application administrator can define:

- the applications that are included in the release
- the library types to include
- the SYSLIB concatenation
- the library types shared between applications in build processing

Release Package

You create a *release package* when you attach a simple change package to a particular release that has a starting area defined. The install date for a release package must fall within the range of the install dates defined for the release. When you attach a package to a release, ERO takes control of building the SYSLIB concatenations for build jobs.

If components in area libraries must be changed, you make the changes in package staging libraries using the familiar ChangeMan ZMF procedures. Package check-in, a subsequent step, will then bring the changed components from a package into the starting area.

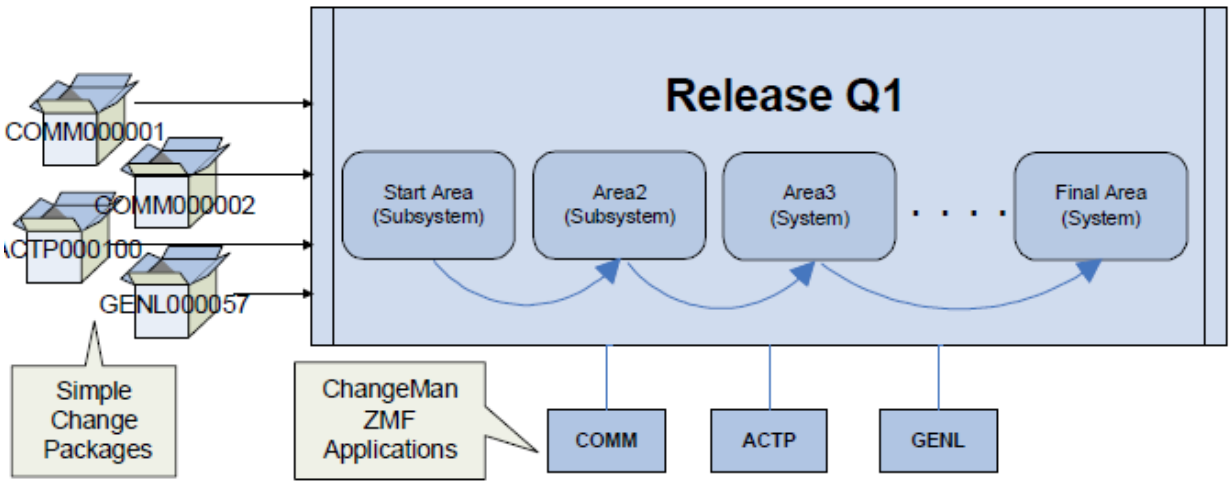

Figure 3-8. Simple change packages are attached to a release.

More information

- Advantages of ERO
- The ERO Release Life Cycle
- Additional ERO Functions

Advantages of ERO

ERO provides comprehensive management for release management:

The software delivery release cycle is predictable.

You can more accurately allocate personnel, such as Testing or Quality Assurance staff, with the establishment of regularly scheduled maintenance and enhancement releases.

Users of the delivered software can more effectively plan for installing and implementing enhancement releases. The development process is enhanced.

The development process is enhanced.

Developers can identify in what release a particular change of a component will be implemented. They can then determine how that change will affect any other releases in motion.

When release schedules shift and features get dropped from a release or added, the release manager can make sure development teams are working with the right components.

Many manual processes can be replaced.

ERO manages all components across all applications that are joined to a release. Release managers can get information about these components in real time. They no longer have the labor-intensive tasks of tracking down and communicating information between teams, or customizing the base product. In addition, you can create custom reports about release component information using the delivered Serena XML Services.

Automatic notification of major release events, like check-in, check-off, install and release approvals, are part of ERO. Customized exits and code are no longer necessary.

There is enhanced functionality over participating packages.

Install date improvements: There is no date verification when you relate participating packages to complex packages. Packages with disparate install dates can end up under the same complex package. With ERO, all packages you attach to a release must have install dates within the release install range. You are ensured the package install dates are valid for that release.

SYSLIB improvements: Once you associate participating packages with a complex package, components in development are included in the SYSLIB concatenations for copybooks and load modules. Errors in compilations and links can be the result, as the components being developed may not be ready for use by other components. With ERO, components are not included in SYSLIB concatenations until they are checked into the release areas.

Blocking areas: With participating packages, release managers need to freeze all the packages to prevent their content from changing. This can become complicated as the packages may not be under the control of the release manager. With ERO, release managers can block entire release areas. When an area is blocked, the contents of the release area are prevented from changing, regardless of the status of any of the release packages.

More Information

Key Terms and Concepts

The ERO Release Life Cycle

Additional ERO Functions

The ERO Release Life Cycle

The ERO release life cycle overlaps the package life cycle used in the base ChangeMan ZMF product. It consists of a number of steps where changed components move through the release areas for synchronizing and testing the changes. The last step installs from the final area into production and ripples the baseline libraries of the release applications.

The following steps trace a release life cycle in ERO:

1. Create a Release and Release Areas

You make ERO administration entries to create a release and its associated areas. These functions are typically performed by ERO release managers and ChangeMan ZMF administrators.

2. Add Install Approval Groups and Area Approvers

You add install approval groups after you create the release. Additional approvals may be dynamically added based on library types included in the release. You can also establish area approvers.

In the following diagram, the approval group DBA is dynamically added as Release Q1 contains DBRM components which reside in a DBR library type. This release will now require DBA approval, along with MGRS and OPER approval.

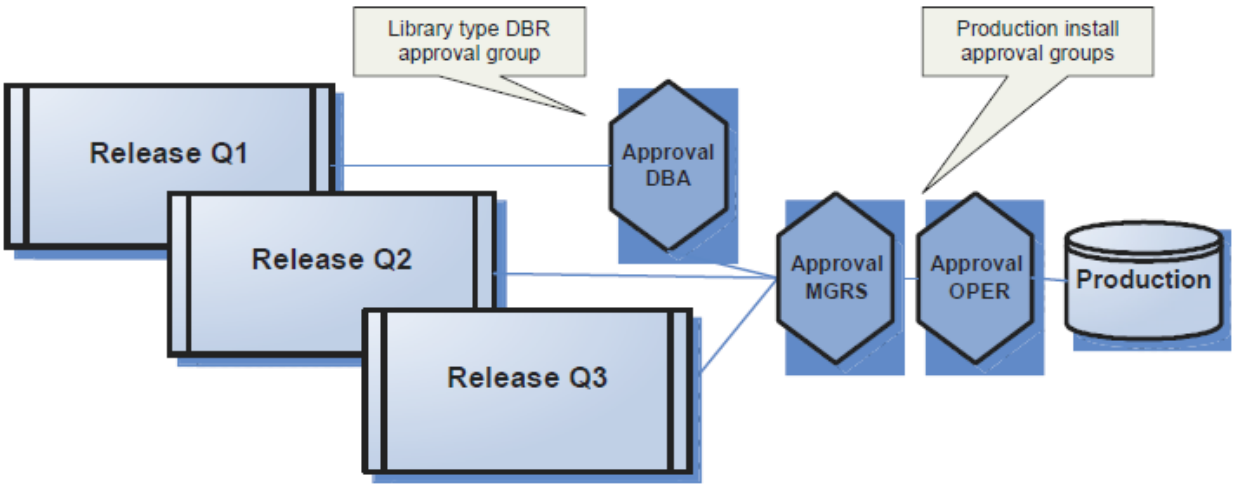

Figure 3-9. Approval groups can be dynamically added for installation.

3. Attach Change Packages to a Release

When you attach a change package to a release, it brings new or changed components into the ERO release life cycle. After you attach a package to a release, the components in the package remain under package control but proceed through the release life cycle. Prior releases' final system libraries and all current release area libraries are included in the SYSLIB concatenation of compiles and links within the package.

4. Approve Area Check-in

The approval process begins with notification to the area check-in approvers. Check-in approval can be used to verify that release areas and applications are reviewed by the appropriate personnel. Check-in approval opens a release area for package or area check-in.

5. Check in a Package

When you check in a package attached to a release, components are brought into the starting area defined for that package. This step begins the integration of the package components with other release components, which are in development in other change packages of that release. With the appropriate rules in place, you audit, freeze, and approve packages before they are checked into a release.

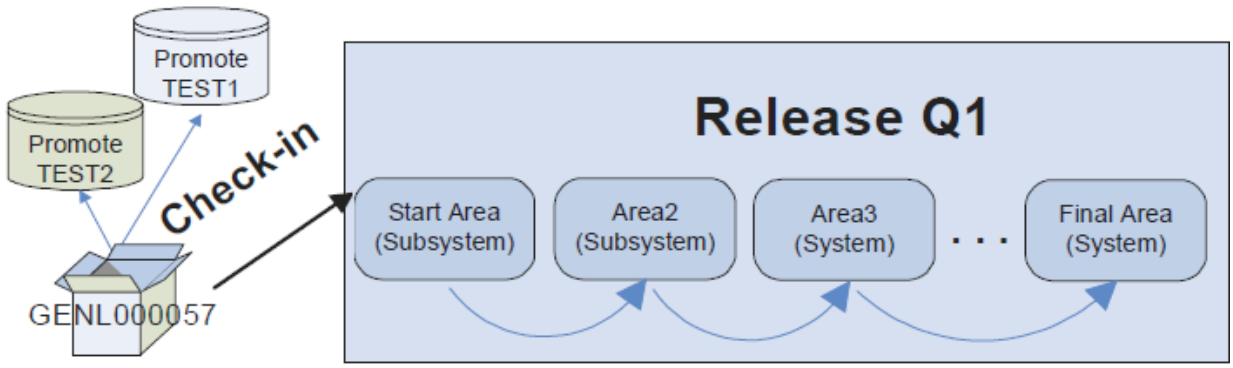

Figure 3-10. Package check-in brings components into the release's starting area.

6. Check-in an Area

You copy components from the libraries for one area into the libraries for another area with area check-in. Release components are advanced through the hierarchy of areas and progressively integrated.

You issue the check-in command on the current area, which copies the components from the current area to the next area. In the following diagram, the check-in command is issued on the Start Area to migrate components to the Unit Test Area.

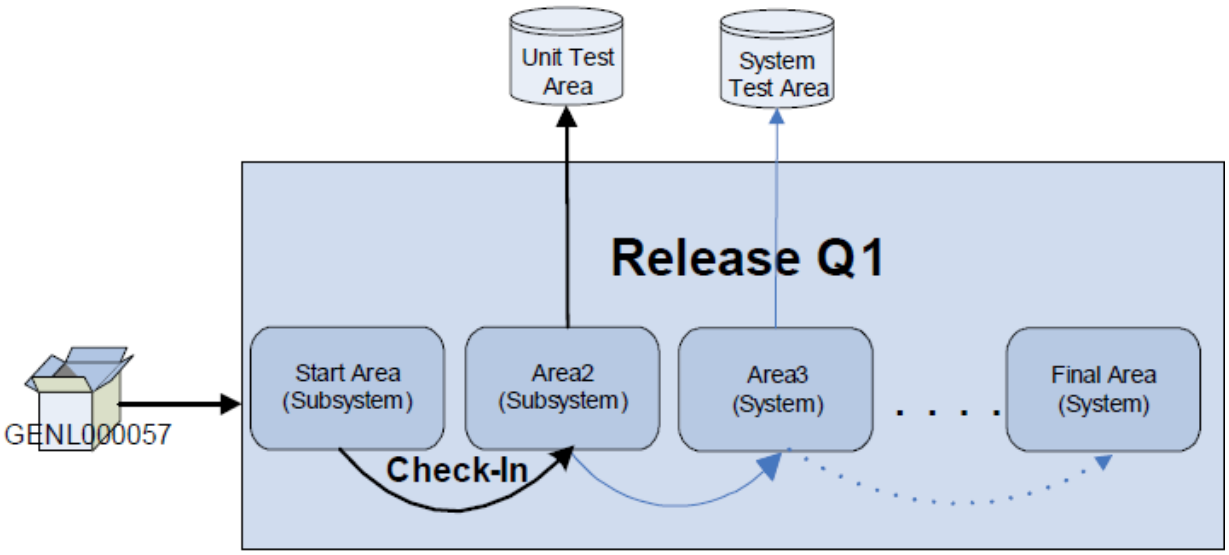

Figure 3-11. Area check-in copies area libraries into libraries of another area.

7. Release Audit an Area

Release audit checks for out-of-sync conditions across releases. For example, if you modify a copybook after you have compiled the source that calls it, release audit will note that condition.

Release audit examines components in:

- libraries for a particular release area
- libraries for other areas in the release
- libraries in prior releases
- baseline libraries

It evaluates relationships between:

- different versions of the same component
- components and other components they include, such as copybooks and statically linked load modules
- link control cards and the associated load modules

The Audit job records audit data and out-of-sync conditions in release audit tables.
Reports can be run against these tables, or queried using XML Services. The Release Audit report produced by the program CMNRARPT includes these sections:

- Directory information for non-load components
- Directory information for load components
- Copybook within Source

- NCAL Loads with composite loads
- Object components within composite loads
- Summary
- Recommendations

Here is a sample report produced by CMNRARPT

```
******************************** TOP OF DATA
********************************************************************************
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 1
**********************************(Release Area Processing)**********************************
*Release Identifier ===> ALICE12 Created: 20150825 Release Install Date ===> 20151201 *
*Area Identifier ===> OUTAREA Area Status ===> BLOCKED *
*Component analysis type ===> Description of Member from Directory Entry for Lib Type-(LOD) *
*************************************************************************************************************************
*------------- P r e v i o u s  V e r s i o n -------------* *------------------ A r e a  L i b r a r y -----------------*
****************************************************** ************************************************************
Name Build Area Changed Release Tso-id Name Build Appl/Pkg# Changed Size VV.MM Tso-ID
________ ____ ________ _____________ ________ ________ ________ ____ __________ _________________ ____ _____ ________
BILLSUB1 1900-01-01 00.00 " BILLSUB1 CHER000169 2015-08-25 06.13 06.01 ACHERNA
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 2
**********************************(Release Area Processing)**********************************
*Release Identifier ===> ALICE12 Created: 20150825 Release Install Date ===> 20151201 *
*Area Identifier ===> OUTAREA Area Status ===> BLOCKED *
*Component analysis type ===> Description of Member from Directory Entry for Lib Type-(LST) *
*************************************************************************************************************************
*------------- P r e v i o u s  V e r s i o n -------------* *------------------ A r e a  L i b r a r y -----------------*
****************************************************** ************************************************************
Name Build Area Changed Release Tso-id Name Build Appl/Pkg# Changed Size VV.MM Tso-ID
________ ____ ________ _____________ ________ ________ ________ ____ __________ _________________ ____ _____ ________
BILLSUB1 2016-02-01 09.19 SERS BILLSUB1 CHER000169 2015-08-25 06.13 06.01 ACHERNA
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 3
**********************************(Release Area Processing)**********************************
*Release Identifier ===> ALICE12 Created: 20150825 Release Install Date ===> 20151201 *
*Area Identifier ===> OUTAREA Area Status ===> BLOCKED *
*Component analysis type ===> Description of Member from Directory Entry for Lib Type-(SRC) *
*************************************************************************************************************************
*------------- P r e v i o u s  V e r s i o n -------------* *------------------ A r e a  L i b r a r y -----------------*
****************************************************** ************************************************************
Name Build Area Changed Release Tso-id Name Build Appl/Pkg# Changed Size VV.MM Tso-ID
________ ____ ________ _____________ ________ ________ ________ ____ __________ _________________ ____ _____ ________
BILLSUB1 2016-02-01 09.19 ACHERNA ERR0417! BILLSUB1 CHER000169 2015-08-25 06.13 06.01 ACHERNA
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 4
**********************************(Release Area Processing)**********************************
*Release Identifier ===> ALICE12 Created: 20150825 Release Install Date ===> 20151201 *
*Area Identifier ===> OUTAREA Area Status ===> BLOCKED *
*Component analysis type ===> Description of Member from Directory Entry for Lib Type-(LOD) *
*************************************************************************************************************************
*------------- V e r s i o n  L i b r a r y --------------* *------------------ A r e a  L i b r a r y -----------------*
****************************************************** ************************************************************
Name Size Linkdate Setssi Build Area Release Name Appl/Pkg# Build Size Linkdate Setssi AC
________ ________ __________ ________ ____ ________ ________ ________ __________ ____ ________ __________ ________ __
NO DIRLOD RECORDS FOUND
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 5
**********************************(Release Area Processing)**********************************
*Release Identifier ===> ALICE12 Created: 20150825 Release Install Date ===> 20151201 *
*Area Identifier ===> OUTAREA Area Status ===> BLOCKED *
*Component analysis type ===> Copybook (CPY) Members within Source (SRC) Code *
*************************************************************************************************************************
*------------- V e r s i o n  L i b r a r y --------------* *------------------ A r e a  L i b r a r y -----------------*
****************************************************** ************************************************************
Copybook Source Copybook Lib
Name Build Area Changed Release Tso-id Name Name Type Build Appl/Pkg# Changed Tso-id
________ ____ ________ _____________ ________ ________ ________ ________ ____ ____ __________ _________________ ________
NO COPS RECORDS FOUND
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 6
**********************************(Release Area Processing)**********************************
*Release Identifier ===> ALICE12 Created: 20150825 Release Install Date ===> 20151201 *
*Area Identifier ===> OUTAREA Area Status ===> BLOCKED *
*Component analysis type ===> NCAL (NCL) Loads within Composite Load Module (LOD) *
```

```
*************************************************************************************************************************************
*------------- V e r s i o n  L i b r a r y --------------* *----------------- A r e a  L i b r a r y -----------------*
******************************************************** ********************************************************************
Called Module Calling Called Lib
Module Size Linkdate Setssi Build Area Release Module Module Type Appl/Pkg# Build Size Linkdate Setssi
________ ________ __________ ________ ____ ________ ________ ________ ________ ____ __________ ____ ________ __________
________
NO CLOD RECORDS FOUND
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 7
*********************************(Release Area Processing)**********************************
*Release Identifier ===> ALICE12 Created: 20150825 Release Install Date ===> 20151201 *
*Area Identifier ===> OUTAREA Area Status ===> BLOCKED *
*Component analysis type ===> Object (OBJ) Components within Composite Load Module (LOD) *
*************************************************************************************************************************************
*------------- V e r s i o n  L i b r a r y --------------* *----------------- A r e a  L i b r a r y -----------------*
******************************************************** ********************************************************************
Object Changed Date Module Calling Object Lib Changed date Module
Name Size /Linkdate Setssi Build Area Release Module Name Type Appl/Pkg# Build Size /Linkdate Setssi
________ ________ __________ ________ ____ ________ ________ ________ ________ ____ __________ ____ ________ __________
________
NO COBJ RECORDS FOUND
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 8
*********************************(Release Area Processing)**********************************
*Release Identifier ===> ALICE12 Created: 20150825 Release Install Date ===> 20151201 *
*Area Identifier ===> OUTAREA Area Status ===> BLOCKED *
*Component analysis type ===> LCT Components which generate named executables *
*************************************************************************************************************************************
*------------- V e r s i o n  L i b r a r y --------------* *----------------- A r e a  L i b r a r y -----------------*
******************************************************** ********************************************************************
Target Module Change/ LCT Target Lib Change/
Module Size Linkdate Setssi Build Area Release Cmponent Module Type Appl/Pkg# Build Size Linkdate Setssi
________ ________ __________ ________ ____ ________ ________ ________ ________ ____ __________ ____ ________ __________
________
NO CLCT RECORDS FOUND
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 9
**************************(Release Area Processing - HFS Components)**************************
*Release Identifier ===> ALICE12 Created: 20150825 Release Install Date ===> 20151201 *
*Area Identifier ===> OUTAREA Area Status ===> BLOCKED *
*Component analysis type ===> Description of Member from Directory Entry for Lib Type-(SRC) *
*************************************************************************************************************************************
*-------------- P r e v i o u s  V e r s i o n ---------------* *---------------- A r e a  L i b r a r y ----------------*
******************************************************** ********************************************************************
Component Name
Release Area Changed ID Appl/Pkg# Changed Size ID
________ ________ ____________________ ________ __________ ____________________ _________ ________
NO DIRSRC HFS RECORDS FOUND
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 10
**************************(Release Area Processing - HFS Components)**************************
*Release Identifier ===> ALICE12 Created: 20150825 Release Install Date ===> 20151201 *
*Area Identifier ===> OUTAREA Area Status ===> BLOCKED *
*Component analysis type ===> Description of Member from Directory Entry for Lib Type-(LOD) *
*************************************************************************************************************************************
*-------------- P r e v i o u s  V e r s i o n ---------------* *---------------- A r e a  L i b r a r y ----------------*
******************************************************** ********************************************************************
Component Name
Timestamp Size Release Area Appl/Pkg# Timestamp Size
____________________ _________ ________ ________ __________ ____________________ _________
NO DIRLOD HFS RECORDS FOUND
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 11
**************************(Release Area Processing - HFS Components)**************************
*Release Identifier ===> ALICE12 Created: 20150825 Release Install Date ===> 20151201 *
*Area Identifier ===> OUTAREA Area Status ===> BLOCKED *
*Subcomponent relationships to libtype-(HFS) *
*************************************************************************************************************************************
*-------------- P r e v i o u s  V e r s i o n ---------------* *---------------- A r e a  L i b r a r y ----------------*
******************************************************** ********************************************************************
>>Component Name -> Subordinate Name
Timestamp Size Area Release Libtype Appl/Pkg# Timestamp Size
____________________ _________ ________ ________ ________ __________ ____________________ _________
NO CLOD HFS RECORDS FOUND
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 12
Legend and Summary Report
The area level of audit chosen at this point is 0 -
Audit is recommended but entirely optional.
Error Conditions Detected:
ERR0417! (VERSION REGRESSION) ===> 0001
Highest return code encountered ===> 12
Area OUTAREA passed the Audit with a return code of 12.
Change Man Release Audit Report Friday April 01,2016 (2016/092) 17:57:04 PAGE: 13
Recommendation Summary Report
```

```
Listed below are some solutions to resolving error situations that have
been detected within this audit report.
ERR0417! THE FIRST OCCURRENCE OF THIS COMPONENT IN THE RELEASE
AREA/BASELINE CONCATENATION (CONSOLIDATED BASELINE) HAS BEEN CHANGED SINCE THE
COMPONENT IN THE AUDITED AREA WAS CHECKED OUT. THESE LATER CHANGES MADE TO THE
CONSOLIDATED BASELINE VERSION OF THIS COMPONENT MAY NOT BE REPRESENTED IN THE
AUDITED AREA.
CHECK-OUT THE UPDATED LATEST VERSION IN MOTION OF THE COMPONENT FROM THE
CONSOLIDATED BASELINE AND MERGE IT WITH THE VERSION IN THE AUDITED AREA.
End of job; RC = 12
Audit Report produced by CMNRARPT dated 07/08/14 07.32.02
****************************** BOTTOM OF DATA
********************************************************************************
```

8. Block an Area

Blocking an area locks the area down to prevent further changes to area components. When an area is blocked, you cannot check-in or retrieve components. The area owner can set rules for blocking, such as requiring an audit before an area is blocked.

If subsequent area changes are necessary, you can unblock the area. Unblocking an area unlocks the area for further changes to area components.

9. Approve Area Check-off

Check-off approval is an administrative function that grants permission to check-in the contents of area libraries to the next area. You begin the process by notifying check-off approvers. The requirement for check-off approval is determined by the area approval rule. A check-off approval signifies successful completion of the area testing as a step of the release life cycle.

10. Block a Release

Blocking a release locks down the release and its areas in preparation for install. All areas in a release must be blocked before a release can be blocked, and all packages attached to the release must be approved. When you attempt to block a release, ERO executes a pre-install test to validate the release and the contents of the final release area.

The installation JCL is built for each package attached to the release when the release is blocked.

11. Approve a Release for Install

After a release is blocked, all install approvers must enter their approvals before the release is installed. When the last approval is entered, the release status is changed to Approve (APR).

12. Install a Release

Packages are installed from their final system area libraries. This differs from the base ChangeMan ZMF product, where package install occurs from the package staging libraries. Each package is installed based on its own install date and time information. Promotion area libraries are cleaned out during the installation process, and package area components are baselined.

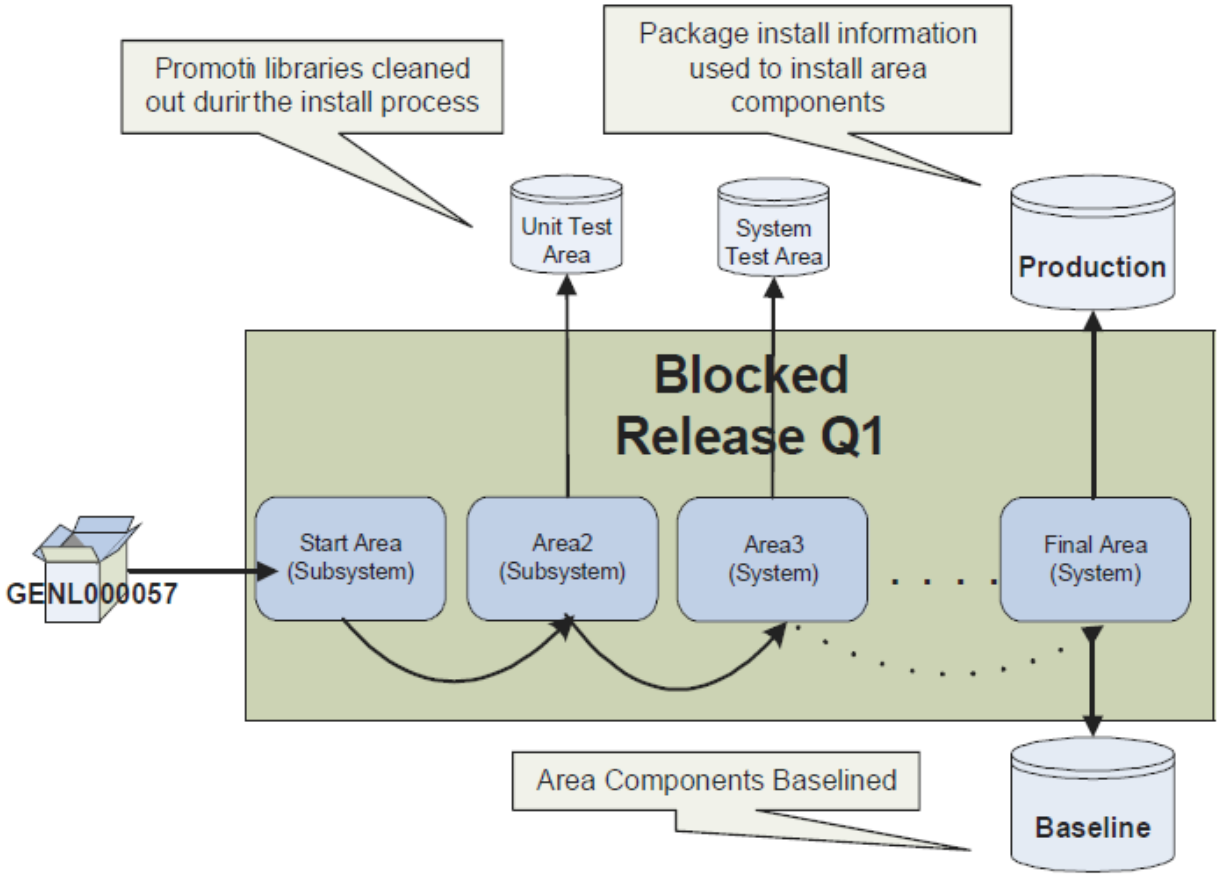

Figure 3-12. Packages are installed from the final system area.

13. (Optional) Back Out a Release

Release backout first verifies that all packages attached to the release are in a state that permits package backout, then submits package backout jobs for the entire release. When complete, the packages and the release are in backout (BAK) status. Both packages and release can then be reverted for further development.

14. (Optional) Revert a Release

When you revert a release, it clears all release install approvals, unblocks the release, and changes the status of the release from approve (APR) or backout (BAK) to development (DEV) status. The status of release areas is not changed, and packages attached to the release are not automatically reverted. This allows you to selectively unblock areas and revert specific packages to rectify the problems before a next attempt to install the release is made.

More information

- Key Terms and Concepts
- Advantages of ERO
- Additional ERO functions

Additional ERO Functions

These functions are available with ERO and can be performed when necessary.

Detaching a Package from a Release

When you detach a package from a release, you sever all relationships to the release, its areas, and area libraries. You break relationships to components in area libraries for that release, and to components in area libraries which identify that release as a prior release.

Retrieving a Package

The retrieve package function removes all package components from the libraries for an area. You must remove package components from area libraries to:

- Detach a package from a release
- Check-in new versions of all package components from a different package

Retrieving from an Area

The retrieve area function removes components from area libraries. You must retrieve components from area libraries to:

- Detach a package from a release
- Check-in a new version of the component from a different package
- Check-in a component by a person different from the person who last checked in the component

Area Promotion

Area promotion populates test environment libraries with components from ERO area libraries. Area demotion removes area components from test libraries.

ERO area promotion is built on standard ChangeMan ZMF services for package promotion. This makes area promotion similar to base ChangeMan ZMF package promotion in some ways and different in other ways. This then makes it possible, for example, for you to:

Use the promotion sites, levels, and libraries that you have configured in application administration in the base product.

Execute special functions like Db2 binds and CICS PHASEIN that you have enabled in promotion skeletons.

As a result, sites are able to design their areas to conform to their promotion environments, i.e. DEV area to one place, UNIT area to another, and so on. Within ERO promotion is on an application level and not a single package like base ZMF. A release package can't inadvertently be promoted to a promotion site/level defined to ERO with base ZMF promotion, as it can only be promoted to non ERO-defined promotion sites/levels.

ERO promotion uses the existing application promotion definitions as well as the libraries. They are assigned to different areas of the release and are part of the release life cycle.

Testing an Area

The test area function compares the contents of an area to the contents of packages attached to the release. Error conditions are displayed online.

As you consolidate package components in a release and as you work in packages to correct errors you find in release testing, you may create mismatches between the contents of areas and the contents of packages attached to the release. When you block a release, ERO automatically executes a test to detect mismatches between the final release area and contents of release packages. You can find these errors earlier in the release life cycle by manually executing the test area function against any release area.

Testing a Release

The test release function executes the test area function against the final area of a release. You cannot block a release until all discrepancies between the final area and packages attached to the release are resolved.

If automatic cleanup is enabled in the release definition, then automatic cleanup is executed in test release the same as it is in test area.

More Information

Roles and Responsibilities

4. Roles and Responsibilities

This chapter describes the responsibilities for the following ERO roles.

- Global Administrator
- Release Manager
- Application Administrator
- Developer

The Role of Global Administrator

Global administrators establish global parameters for the release environment. Their responsibilities include:

Deciding Global Configuration

The global administrator, working with the release manager, determines:

- The DSN high-level qualifier used during the dynamic allocation of release area libraries.
- The HFS high level directory used during the dynamic allocation of release area paths and directories.
- The dataset pattern for the naming convention to be used for these libraries.
- The minimum audit level that can be set for all releases.

The following panel (CMNRMGA1) shows the Global Parameters of High Level Qualifier and Dataset Pattern (=A.R.G.1).

```
CMNRMGA1 Global Parameters
Command ===>
High level qualifier . . . . . .
High level HFS pathname . . . . . .
Dataset pattern . . . . . . . . . . HRAPL (Default is HRAPL)
                                   H - High Level Qualifier
                                   R - Release Name
                                   A - Release Area Name
                                   P - Release Application Name
                                   L - Release Library Type
                                       (must be last in pattern)
Minimum Audit Level for all Releases . . . . 0 (0,1,2,3,4,5)
```

Deciding Approvers

All approvers are defined at the global level. When you subsequently build a release, you select approvers from the global approver list. The following panel (CMNRMAPL) shows a list of global approvers, and uses the CREATE command is to create an approver to add to this list. (=A.R.G.2)

```
CMNRMAPL Global Approver List Global Approver Created
Command ===> Scroll ===> CSR

   Security entity  Order no.  Description
__ ACCTPAY          010        Accounts Payable Approver Manager
__ ACTPLEAD         010        Lead Developer ACTP Application
__ CIO              010        Chief Information Officer
__ FINACCTG         010        Financial Accounting Manager
__ GENLEDGR         010        General Ledger Manager
__ GENLLEAD         010        Lead Developer - GENL Application
__ SYSDVMGR         010        System Development Manager
__ OPS              020        Data Center Operations
__ DBA              030        Database Administrator
****************************** Bottom of data ********************************
```

Approvers are automatically notified at certain checkpoints in the release life cycle. The following notifications are sent:

- INSTALL APPROVER notification indicates approval is required for a release to be installed.
- CHECK-IN APPROVER notification indicates approval is required for a release or release package to be checked into an area.
- CHECK-OFF APPROVER notification indicates approval is required to signify a release is ready to be checked into the next area.
- ASSOCIATED APPROVER notification indicates approval is required, as components for the library type which requires their approval became part of the release.

More information

Roles and Responsibilities

The Role of Release Manager

ERO introduces the role of a Release Manager. This role creates and manages releases and their associated areas. The responsibilities of a release manager include:

Creating Releases

You create releases with a unique 1-8 character release identifier. The release identifier is used as a node in the dataset names for release area libraries. Information required includes:

- Release Description
- Audit Level
- The minimum rules that may be used for areas in the release. The least restrictive rules can be set for Approvals, Blocking, Check-in, and Retrieve.
- Ascending or Descending SYSLIB Order of Area libraries

  Ascending order is forward from the current area libraries to the final area libraries. Ascending order ensures you that the latest changes checked into areas starting from the target area to the final area are used instead of earlier changes that are resident in the final area.

  Descending order is backward from the final area libraries to the current area libraries. Descending order ensures you that the earliest changes checked into areas starting from the final area to the target area are used instead of later changes.
- Auto Clean-up of Release Packages When you test the content of the final release area against the content of the release packages, components in the staging libraries that have not made it into the final area can automatically be deleted.
- Install Date and Time Range
- Installation/Problem Handling Instructions
- Release Order

The following panel shows the first part of release parameters you enter when you define a release. The input fields include, among others, Release Description, Minimum rule levels, and SYSLIB Concatenation Order.

```
CMNRMRC0 FIN6460 Release Management Parameters - Part 1 of 2
Command ===>

Release description . . . . . FIN6450 Release for test
Creator . . . . . . . . . . . USER123
Creator's Phone Number . . . 11292
Work request . . . . . . . . WR 9015
Department . . . . . . . . . FINANCE
Minimum audit level . . . . . 0   (0,1,2,3,4,5)
Minimum approval rule . . . . 0   (0,1,2,3)
Minimum blocking rule . . . . 0   (0,1,2,3,4,5,6,7)
Minimum Check-in rule . . . . 0   (0,1,2,3,4,5,6,7)
Minimum retrieve rule . . . . 0   (0,1,2,3)
SYSLIB concatenation order .  A   (A-Ascending,D-Descending)
Default IHA audit setting . . N   (Y/N/C)
Enter "/" to select option
  __ / Enforce IHA default setting
  __ Bypass checkin package requirements
  __ Allow empty packages to process in release
  __ Auto cleanup of packages in DEV status
  __ Auto cleanup of packages in FRZ status
  __ Auto cleanup of packages in APR status
```

Defining Install Approvers for a Release

You can select which approvers are needed for a particular release from the list of approvers defined at a global level.

Defining Areas to a Release

You create areas with a unique 1-8 character area name, which is used in the dataset name for release area libraries. At least one start (subsystem area) and one final (system area) must be defined in a release.

Required information includes:

- Area Description
- Audit level
- Any Prior Area Name
- The Next Area Name
- The minimum rules that may be used for areas in the release. The least restrictive rules can be set for Approvals, Blocking, Check-in, and Retrieve. These Area Functions' Rules can be more restrictive than the Release minimum rules.

The following panel shows the above parameters as input fields when creating an area.

```
CMNRMAC0 FIN6410 R041218 Area Parameters - Part 1 of 2
Command ===>

Area description . . Final area for Finance components
Area step number . . . . . 0003
Area step type . . . . . 1   (Subsystem-0 or System-1)
Any prior area name . . . . . GENLEDGR
The next area name . . . . . .
Area audit level . . . . . 0   (0,1,2,3,4,5)
Area approval rule . . . . . 0   (0,1,2,3)
Area blocking rule . . . . . 0   (0,1,2,3,4,5,6,7)
Blocking entity . . . . . .      (Entity Name)
Area check-in rule . . . . . 0   (0,1,2,3,4,5,6,7)
Check-in Entity . . . . . .      (Entity Name)
Area retrieve rule . . . . . 0   (0,1,2,3)
Retrieve entity . . . . . .      (Entity Name)
Enter "/" to select option
  __ / Allow component checkout
  __ / Add associated approvers
  __ Exclude area from SYSLIB
  __ Override overlaid components
```

```
CMNRMAC1 FIN6410 R041218 Area Parameters - Part 2 of 2
Command ===>

Enter "/" to select option
  __ Exclude packages in DEV status
  __ Exclude packages in FRZ status
  __ Exclude packages in APR status
  __ Exclude empty packages
  __ Exclude package integrity check
```

Adding Check-In and Check-Off Area Approvers

You can add area check-in and check-off area approvers, which are selected from the Global Approval List. If specified, check-in approval is required before check-in to the area is permitted. Check-off approval is required before migration from this area.

The following panels show part 1 and 2 for defining area approvers. Input fields are Approver Order Number, Check-in Approver, Check-off Approver, and Description.

```
CMNRMAA0 Approver Parameters - Part 1 of 2
Command ===>

Release: FIN6410    Area: ACCTPAY   Entity: ACTPLEAD
Description . . . . . . . . Lead Developer ACTP Application
Order Number . . . . . . . .  0010
Enter "/" to select option
  __ Check-in Approver
  /  Check-off Approver
Approver List Count . . . . 0001
```

```
CMNRMAA1 Approver Parameters - Part 2 of 2 Row 1 to 1 of 1
Command ===> Scroll ===> CSR

Release: FIN6410    Area: ACCTPAY   Entity: ACTPLEAD
Approver: Lead Developer ACTP Application
Order no: 0010
Vehicle User(s) to notify
MVSSEND jsmith
```

Blocking a Release

When you block a release, it locks down the release and its areas in preparation for install.

The following panel lists releases, and indicates the line command BK is used to block a release. Other available line commands include AR, which is used to approve a release, and BR, which is used to view backout reasons.

```
CMNRMRLF Release List Row 1 to 2 of 2
Command ===> Scroll ===> CSR

Release Sta Install   Work request  Dept    Aud Creator Pkgs
FIN6410 DEV 20160428  WR 9010       FINANCE     JSMITH  00002
FIN6430 DEV 20160428  WR 9030       FINANCE     JSMITH  00002
```

Approving a Release

After a release is blocked, all install approvers must enter their approvals before the release will install. If a release is unblocked, all approvals entered up to that point are cleared, and they must be entered again after the release is blocked. If an approver rejects the release, the release must be reverted to clear the rejection, and all install approvals must be entered again.

Installing a Release

Packages attached to a release are distributed when all release approvals are entered and the release status is changed to approved (APR). Each package attached to a release is installed according to the package Scheduler and Install Date/Time. When a package attached to a release is installed successfully, the package status in the development instance is changed to baseline (BAS).

Although release components are installed on a package-by-package basis, the components are copied from final release area libraries rather than from package staging libraries.

Backing Out a Release

Release backout submits backout jobs for all packages attached to the release. After completion, the release status is changed to BAK. To return the release to DEV status, you would revert the release.

Unblocking a Release

Unblocking a release unlocks the release for further changes. Unblocking a release does not unblock the areas in the release. You must unblock release areas to change release components.

If a release is unblocked, all approvals entered up to that point are cleared, and they must be entered again after the release is blocked.

Unblocking an Area

Release managers can block area libraries to disallow component migration into predefined release areas. They unblock areas libraries to allow component migration to begin.

More information

Roles and Responsibilities

The Role of Application Administrator

Application Administrators manage which applications join a release.

Joining Applications to a Release

Joining applications to a release helps manage if, when, and with what release an application will be updated. Before an application can join in a release, you must define areas to the release.

The following panel shows the ACTP application has joined a release. The other applications that are listed can be joined to the release by using the S line command.

```
CMNRMJAP Join FIN6410 Application Selectio Application Joined
Command ===> Scroll ===> CSR

Appl Status Application Description
JHFS *Joined* JHFS hfs only application
****************************** Bottom of data ******************************
...
+-------------------------------------------------------------------+
| CMR0527A - Application JHFS has been successfully joined release. |
+-------------------------------------------------------------------+
```

Selecting Library Types

You select which library types in the joined application are included in the release.

You can build special purpose releases by omitting some library types from the application joined to the release. For example, you can create a release for on-line components by omitting library types for batch components from all applications joined to the release. If you then attempt to check-in a package that contains batch components, the batch components will be disallowed from check-in, and the release cannot be blocked.

The following screen shows a list of library types. With the S line command, you select which library types in the joined application are included in a release.

```
CMNRMLAL     ACTP Library Selection List  Row 1 to 15 of 15
Command ===>                              Scroll ===> CSR

  Lib
  Type Request Description
_ ACT _________ Component Activity File_____________________
_ CPS _________ Copybooks with same name, different library_
_ CPY _________ Copybooks for ACTP Application______________
_ CTL _________ Job Control Statements (x)__________________
_ DBB _________ Db2 BIND PLAN Command Members_______________
_ DBR _________ Db2 DBRM____________________________________
_ JCL _________ Execution JCL_______________________________
_ LCT _________ Link Edit Control Statements________________
_ LOD _________ Executable Load Modules_____________________
_ LOS _________ Load for Subprograms to be Statically Linked
_ LST _________ Compressed Stage Listings___________________
_ PKG _________ Db2 BIND PACKAGE Command Members____________
_ PRC _________ Cataloged Procedures________________________
_ SRC _________ Source for programs to be Linked Executable_
_ SRS _________ Source for Subpgms to be Statically Linked__
****************************** Bottom of data ********************************
```

Defining SYSLIB Concatenations

ERO allows you to define SYSLIB concatenations, which are based on:

- application
- library type
- language
- procedures

These SYSLIB concatentations are used for build processes to determine which components will be used for copybook and load module inclusion. The release application library types eligible for SYSLIB definition are like-source, like-load, and like-linkcontrol library types defined to the joined application.

The following panel shows libraries for type CPY will be concatenated over libraries for type CPS. This is in application ACTP for library type SRC, with language COBOL2 and procedure CMNCOB2.

```
CMNRMSY1  ACTP SRC SYSLIB LIBRARY ORDER   Row 1 to 2 of 2
Command ===>                              Scroll ===> CSR

Library Type: SRC   Language: COBOL2  Procedure: CMNCOB2
Library   Order
type      number
_ CPS     00000
_ CPY     00000
****************************** Bottom of data ******************************
```

More information

Roles and Responsibilities

The Role of the Developer

As a developer, you work at a component level to ensure the correct changes are made to components belonging to the correct release. Typical functions include:

Attaching Packages to a Release

You attach a change package to a release to bring components you are changing into the ERO release life cycle. The components in your package remain under package control, and you execute standard change package life-cycle functions to prepare these components for installation into production.

The following screen shows Package Create parameters. Set ATTACH PACKAGE TO RELEASE to YES at package creation or package update to attach a package to an ERO release. There is no ERO function to attach a package.

```
CMNCRT0R                    Create: Create a New Package
Option ===> ________________________________________________

        L Long method                   S Short method
        D No package description        I No installation instructions

Package title
Demo package ____________________________________________
Application . . . . . .  DEMO     (Blank or pattern for list)
Requester's name . . . . . John Doe_________________
Requester's phone . . . . .  (555) 555-5555________
Work request . . . . . 100001000106________
Department . . . . . IDD___
Package level . . . . . .  1_          (1. Simple 2. Complex
                                        3. Super 4. Participating)
Package type . . . . . PLANNED__   (Planned or Unplanned)
Package time span . . . . .  PERM__      (Permanent or Temporary)
Package to copy forward . .  _________   (Optional package name)
Unplanned reason code . . .  ___         (* for list)
Temporary change duration .  ___         (In days)
Notify user . . . . . .  JDOE__

Enter "/" to select option
 __ Attach package to release
```

Checking Out Components

After you attach a package to a release, you can check out components into your package from releases that have not been installed yet. Checkout from release lets you start coding from a version of a component that is more recent than the version in baseline, and which already contains earlier changes from your project or another project.

If you check out a version of a component from a prior release, you may be able to avoid a regression out-of-sync audit error in your release after the prior release is installed. If you check out a component from an area in the current release, you will eventually encounter an overlay condition in package or area check-in unless the other version is retrieved.

Checking In Packages

When you check in a package, you bring components from that package attached to a release into the starting subsystem area defined for that package. This step begins the integration of your package components with other release components in development in other change packages.

Package check-in initiates these activities:

- Allocates area libraries for all areas in the release for the library types that are contained in the package
- Populates starting release area libraries

- Makes the components available to build processes in other packages in the same application that are attached to the release
- Makes the components available to build processes in packages in other applications that have this application defined as a related application
- Starts the process of resolving multiple versions of the same component, as only one version of the component can reside in the same area

With appropriate area rules in place, you audit, freeze, and approve packages before they are checked into a release's starting area.

Checking In an Area

Area check-in allows you to copy components from the libraries for one area into the libraries for another area. Check-in advances release components through the hierarchy of areas that progressively integrate release components.

Area check-in initiates the following:

- Populates the area application libraries for the next area defined for the release
- Makes the components available to build processes in other packages in the same application that are attached to the release
- Makes the components available to build processes in other packages in the same application that define this release as a prior release
- Makes the components available to build processes in packages in other applications if this application is defined as a related application
- Continues the process of squeezing out multiple versions of the same component, as only one version of the component can reside in the same area

Generating an Area

Generating an area allows you to initiate build processing for every source and load component in the area libraries.

Auditing an Area

Release audit examines the components in libraries for a particular release area, as well as libraries for other areas in the release, libraries in prior releases that will be installed sooner, and baseline libraries. It evaluates relationships between different versions of the same component, and it evaluates relationships between components and other components they include like copybooks and statically linked load modules.

More information

Roles and Responsibilities

5. Legal Notice

For information about legal notices, trademarks, disclaimers, warranties, export and other use restrictions, U.S. Government rights, patent policy, and FIPS compliance, see https://www.microfocus.com/about/legal/.

The only warranties for products and services of Micro Focus and its affiliates and licensors ("Micro Focus") are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Third-Party Notices

Additional third-party notices, including copyrights and software license texts, can be found in a 'thirdpartynotices' file in the root directory of the software.

Specific notices

In accordance with the GNU General Public License version 2 with Classpath Exception, you are entitled to the complete OpenJDK source code that went into the JRE used by this product which includes the source code for 3 subclasses of that standard OpenJDK; MultipleGradientPaint, MultipleGradientPaintContext and TypeResolver. Please contact product support if you wish to obtain the source code. This source code will be available for 3 years from the general availability date for version 17.0 SP1.