



ChangeMan[®] ZMF 8.1.4

Customization Guide

Proprietary and Confidential Information

Copyright © 2001–2018 Serena Software, Inc., a Micro Focus Company. All rights reserved.

This document, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by such license, no part of this publication may be reproduced, photocopied, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Serena. Any reproduction of such software product user documentation, regardless of whether the documentation is reproduced in whole or in part, must be accompanied by this copyright statement in its entirety, without modification.

This document contains proprietary and confidential information, and no reproduction or dissemination of any information contained herein is allowed without the express permission of Serena Software.

The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Serena. Serena assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

Trademarks

Serena, TeamTrack, StarTool, PVCS, Comparex, Dimensions, Prototype Composer, Mariner and ChangeMan are registered trademarks of Serena Software, Inc. The Serena logo, Version Manager and Mover are trademarks of Serena Software, Inc. All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

U.S. Government Rights

Any Software product acquired by Licensee under this Agreement for or on behalf of the U.S. Government, its agencies and instrumentalities is "commercial software" as defined by the FAR. Use, duplication, and disclosure by the U.S. Government is subject to the restrictions set forth in the license under which the Software was acquired. The manufacturer is Serena Software, Inc., 2345 NW Amberbrook Drive, Suite 200, Hillsboro, OR 97006 USA.

Publication date: January 2018

Table of Contents

| | | |
|------------------|--|-----------|
| | Welcome to ChangeMan® ZMF | 9 |
| | Guide to ChangeMan ZMF Documentation | 9 |
| | ChangeMan ZMF Documentation Suite | 9 |
| | Using the Manuals | 12 |
| | Searching the ChangeMan ZMF Documentation Suite | 12 |
| | Using Online Help | 13 |
| | Online Tutorial | 13 |
| | Online Help Screens | 13 |
| | Online Error Messages | 13 |
| | Typographical Conventions | 13 |
| | Notes | 14 |
| <i>Chapter 1</i> | Introduction | 15 |
| | Preserving Vendor Versions of ChangeMan ZMF Components | 16 |
| | Using ChangeMan ZMF To Manage ChangeMan ZMF Components | 16 |
| | Nomenclature | 16 |
| <i>Chapter 2</i> | ISPF Skeletons | 19 |
| | Introduction | 20 |
| | Skeleton File Tailoring in ChangeMan ZMF | 20 |
| | Skeleton Naming Conventions | 20 |
| | Skeleton Variables | 21 |
| | Skeleton Imbedding | 22 |
| | Skeleton Maintenance Facility | 23 |
| | Developing Skeletons With File Tailoring Assistance | 23 |
| | Editing Skeletons in File Tailoring Assistance | 25 |
| | Syntax Checking in File Tailoring Assistance | 27 |
| | Debugging Skeletons in Started Task Procedures | 28 |
| | ISPF Table CMNTBN | 31 |
| | Error Codes | 31 |
| | CMN\$\$JBL - JOBLIB / STEPLIB | 31 |
| | Setting Build Parameters | 33 |
| | Build Parameter ISPF Variables | 33 |
| | Build Parameter Skeleton Architecture | 34 |
| | Customization Steps | 35 |
| | Transmit Selected Remote Promote Components | 36 |
| | JES Node Names and Transmission Site Names | 36 |
| <i>Chapter 3</i> | User Exits | 39 |
| | Introduction | 40 |
| | User Exit Source | 40 |

| | |
|--|----|
| User Exit Interface Data | 40 |
| No Access to TCA | 41 |
| Customizing Exits | 41 |
| Find the Exit You Want. | 41 |
| Modify Exit Source. | 41 |
| Assemble Exit Source | 42 |
| Refresh Exit Load | 43 |
| Exits Listed in SYSPRINT | 44 |
| Calling XML Services from User Exits | 44 |
| Exit Descriptions | 45 |
| SEREX001 | 45 |
| SEREX002 | 45 |
| SEREX003 | 45 |
| SEREX005 | 46 |
| CMNEXINS | 46 |
| CMNEX001 | 47 |
| CMNEX002 | 47 |
| CMNEX003 | 47 |
| CMNEX004 | 48 |
| CMNEX005 | 48 |
| CMNEX006 | 48 |
| CMNEX007 | 48 |
| CMNEX008 | 48 |
| CMNEX009 | 49 |
| CMNEX010 | 50 |
| CMNEX011 | 50 |
| CMNEX012 | 50 |
| CMNEX014 | 50 |
| CMNEX015 | 51 |
| CMNEX016 | 51 |
| CMNEX019 | 51 |
| CMNEX020 | 52 |
| CMNEX021 | 52 |
| CMNEX022 | 52 |
| CMNEX023 | 52 |
| CMNEX024 | 53 |
| CMNEX025 | 53 |
| CMNEX026 | 53 |
| CMNEX027 | 53 |
| CMNEX028 | 54 |
| CMNEX030 | 54 |
| CMNEX031 | 54 |
| CMNEX032 | 54 |
| CMNEX033 | 54 |
| CMNEX034 | 55 |
| CMNEX035 | 55 |
| CMNEX036 | 55 |
| CMNEX037 | 55 |

| | | |
|------------------|--|-----------|
| | CMNEX038 | 56 |
| | CMNEX039 | 56 |
| | CMNEX040 | 56 |
| | CMNEX041 | 56 |
| | CMNEX042 | 56 |
| | CMNEX043 | 57 |
| | CMNEX044 | 57 |
| | CMNEX093 | 57 |
| | CMNEX101 | 57 |
| | CMNEX102 | 57 |
| | CMNEX103 | 57 |
| | CMNEX201 | 58 |
| | CMNEX210 | 58 |
| | CMNEX220 | 58 |
| Chapter 4 | User Data | 59 |
| | Package User Information | 60 |
| | Package User Information Field Names | 60 |
| | Package User Information Input Panels | 61 |
| | Package User Information and Exits | 62 |
| | Implementing the Package User Information Facility | 62 |
| | Staging User Options | 64 |
| | User Options Field Names. | 65 |
| | User Option Input Panels | 66 |
| | User Options and Exits. | 68 |
| | Implementing the User Options. | 68 |
| | User Option Example | 69 |
| | Release ID Variables | 71 |
| | Accessing Maintain Release ID Variables | 71 |
| | Creating a New Release ID | 72 |
| | Maintaining an Existing Release ID | 74 |
| | Associating a Release ID with an Application | 76 |
| | Custom V01-V10 Variables | 77 |
| | Custom V01-V10 Field Names. | 78 |
| | Using Custom V01-V10 Variables | 78 |
| | Summary | 79 |
| Chapter 5 | Utilities | 81 |
| | CMNBAHST - Initial History Record | 82 |
| | CMNBAHST Input | 82 |
| | Output | 82 |
| | Sample JCL | 82 |
| | DD Statements | 82 |
| | PARM Options | 83 |
| | SYSIN Parameters. | 83 |
| | Return Codes and Error Messages | 83 |
| | Reporting | 83 |
| | CMNBAT90 - Register Build Output Modules | 83 |

| | |
|---|-----|
| CMNBAT90 Input | 84 |
| Output | 84 |
| Sample JCL | 84 |
| DD Statements | 86 |
| Program Execution Parameters | 86 |
| SYSIN Keyword Statements | 87 |
| Return Codes and Error Messages | 89 |
| Reporting | 89 |
| CMNBAT90 Notes | 89 |
| CMNBAT90 Example - Composite Load Module | 91 |
| CMNBKRST - VSAM MASTER UNLOAD, RECOVER, LOAD | 94 |
| Program Execution Parameters | 94 |
| CMNBKRST Input and Output | 94 |
| Sample JCL | 95 |
| DD Statements | 96 |
| SYSIN Keyword Statements | 96 |
| Return Codes, Completion Codes, and Error Messages. | 96 |
| Reporting | 97 |
| CMNBKRST Notes | 98 |
| CMNCICS1 - CICS NEWCOPY. | 98 |
| CMNCICS1 Input | 100 |
| Output | 100 |
| Sample JCL | 100 |
| DD Statements | 101 |
| PARM Options | 101 |
| SYSIN Parameters. | 102 |
| Return Codes and Error Messages | 104 |
| Reporting | 105 |
| Notes and Comments | 105 |
| CMNCICS6 - CICS CSD Extract | 106 |
| Export Option | 107 |
| Basic Format of CMNCICS6 Export Control Statement | 108 |
| Import Option. | 108 |
| Basic Format of CMNCICS6 Import Control Statement | 109 |
| CICS Keywords processed by CMNCICS6 | 110 |
| CEDA Language Review | 112 |
| CMNFI XMN - Generate SETSSI Data. | 112 |
| Input | 113 |
| Output | 113 |
| Sample JCL | 113 |
| DD Statements | 113 |
| PARM Options | 113 |
| Return Codes and Error Messages | 114 |
| Reporting | 114 |
| CMNIALD0 - Impact Analysis DB2 Load | 115 |
| CMNIALD0 Input | 115 |
| Output | 115 |
| Sample JCL | 115 |

| | |
|---|-----|
| DD Statements | 116 |
| PARM Options | 116 |
| Return Codes and Error Messages | 116 |
| Reporting | 116 |
| Notes or Comments | 116 |
| CMNPMLOD - Master File XML Extractor | 117 |
| CMNPMLOD Input | 117 |
| Output | 118 |
| Sample JCL | 118 |
| DD Statements | 119 |
| PARM Options | 121 |
| No PARM parameter is required in the EXEC statement for CMNPMLOD. | 121 |
| Return Codes and Error Messages | 121 |
| Reporting | 122 |
| Sample CMNPMLOD Extract | 122 |
| Notes or Comments | 122 |
| CMNSSIDN - LINK EDIT Control Preparation | 123 |
| CMNSSIDN Input. | 124 |
| Output | 124 |
| Sample JCL | 124 |
| DD Statements | 124 |
| Program Execution Parameters | 125 |
| SYSIN Control Statements | 126 |
| INCLIB and CMNSSIDN | 127 |
| Return Codes and Error Messages | 128 |
| Reporting | 128 |
| CMNSSIDN Examples. | 129 |
| CMNUPDAT - Stacked Reverse Delta Management | 129 |
| CMNUPDAT Input and Output | 130 |
| Sample JCL | 131 |
| DD Statements | 131 |
| PARM Options | 132 |
| Notes or Comments. | 133 |
| CMNWRITE - Copy And Include Management | 135 |
| CMNWRITE Input | 135 |
| Output | 135 |
| Sample JCL | 136 |
| DD Statements | 136 |
| PARM Options | 137 |
| SYSIN Parameters. | 137 |
| Return Codes and Error Messages | 141 |
| Reporting | 141 |
| Notes. | 142 |
| SERCOPY - Copy Utility. | 144 |
| SERCOPY Input | 145 |
| Output | 145 |
| Sample JCL | 145 |
| DD Statements | 145 |

| | | |
|--|--|-----|
| | PARM Options | 146 |
| | SYSIN Parameters | 147 |
| | Return Codes and Error Messages | 148 |
| | Comments | 148 |
| | Reporting | 149 |
| | SERPRINT - SYSOUT Compression Facility | 149 |
| | Browsing Compressed Listings | 151 |

Chapter 6

| | | |
|--|--|------------|
| | Reports | 153 |
| | Overview of Online Report Generation | 154 |
| | Submitting a Batch Job to Generate a Report | 155 |
| | Analysis of a Sample REXX Reporting Program | 155 |
| | Introductory Comments | 156 |
| | Mainline Program Logic | 157 |
| | Getting User Input. | 158 |
| | Validating User Input. | 158 |
| | Initializing Variables | 160 |
| | Setting Up the XML Service Call | 161 |
| | Calling the Target XML Service | 161 |
| | Diagnosing Errors and Formatting Report Output | 162 |
| | Disconnecting from ChangeMan ZMF | 163 |
| | XML Services Called in Reporting Programs. | 163 |

Appendix A

| | | |
|--|--|------------|
| | Installation Jobs and Transaction Codes | 165 |
| | X Node Data Sets | 166 |
| | Installation Jobs | 167 |
| | Other CMNBATCH Transaction Codes | 170 |

Appendix B

| | | |
|--|--|------------|
| | Skeleton Hierarchy | 171 |
| | Introduction | 172 |
| | Analyzing Skeleton Imbeds | 172 |
| | Standard ChangeMan ZMF Skeletons | 174 |
| | Checkout Skeletons | 174 |
| | Stage Skeletons | 175 |
| | Promotion Skeletons | 183 |
| | Installation Skeletons | 185 |
| | Utility And Other Skeletons | 198 |
| | IMS Option Skeletons | 199 |
| | IMS Stage Skeletons | 199 |
| | IMS Promotion Skeletons | 202 |
| | IMS Installation Skeletons | 204 |
| | IMS Sample Skeletons | 211 |

| | | |
|--|-----------------------|------------|
| | Index. | 213 |
|--|-----------------------|------------|

Welcome to ChangeMan[®] ZMF

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 The *ChangeMan ZMF Customization Guide* provides detailed information about ChangeMan ZMF components that you can modify for functions you need for software change management at your company.
- Audience This document is intended for ChangeMan ZMF administrators and technical staff who are responsible for the installation and maintenance of ChangeMan ZMF software. This document assumes that you are familiar with ChangeMan ZMF functions and architecture.
- Change Bars Change bars in the left margin identify substantive changes in this publication since ChangeMan ZMF release 8.1

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 in PDF format.

| Manual | Description |
|---|---|
| <i>Administrator's Guide</i> | Describes ChangeMan ZMF features and functions with instructions for choosing options and configuring global and application administration parameters. |
| <i>ChangeMan ZMF Quick Reference</i> | Provides a summary of the commands you use to perform the major functions in the ChangeMan ZMF package life cycle. |
| <i>Customization Guide</i> | Provides information about ChangeMan ZMF skeletons, exits, and utility programs that will help you to customize the base product to fit your needs. |
| <i>Db2 Option Getting Started Guide</i> | Describes how to install and use the DB2 Option of ChangeMan ZMF to manage changes to DB2 components. |
| <i>ERO Concepts</i> | Discusses the concepts of the ERO Option of ChangeMan ZMF for managing releases containing change packages. |
| <i>ERO Getting Started Guide</i> | Explains how to install and use the ERO Option of ChangeMan ZMF to manage releases containing change packages. |

| Manual | Description |
|---|--|
| <i>ERO Messages</i> | Describes system messages and codes produced by ChangeMan ZMF ERO. |
| <i>ERO XML Services User's Guide</i> | Documents ERO functions and services available for general customer use. These services are also known as the "green" services and provide mostly search and query functions. |
| <i>High-Level Language Functional Exits Getting Started Guide</i> | Provides instructions for implementing and using High-Level Language (Cobol, PL/1, and REXX) exits, driven consistently by all clients to enforce local business rules in ZMF functions. |
| <i>IMS Option Getting Started Guide</i> | Provides instructions for implementing and using the IMS Option of ChangeMan ZMF to manage changes to IMS components. |
| <i>INFO Option Getting Started Guide</i> | Describes two methods by which ChangeMan ZMF can communicate with other applications: <ul style="list-style-type: none"> ■ Through a VSAM interface file. ■ Through the Tivoli Information Management for z/OS product from IBM. |
| <i>Installation Guide</i> | 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. |
| <i>Java / zFS Getting Started Guide</i> | Provides information about using ZMF to manage application components stored in USS file systems, especially Java application components. |
| <i>Load Balancing Option Getting Started Guide</i> | Explains how to install and use the Load Balancing Option of ChangeMan ZMF to connect to a ChangeMan ZMF instance from another CPU or MVS image. |
| <i>M+R Getting Started Guide</i> | Explains how to install and use the M+R Option of ChangeMan ZMF to consolidate multiple versions of source code and other text components. |
| <i>M+R Quick Reference</i> | Provides a summary of M+R Option commands in a handy pamphlet format. |
| <i>Messages</i> | Explains messages issued by ChangeMan ZMF, SERNET, and System Software Manager (SSM) used for the Staging Versions feature of ChangeMan ZMF. |
| <i>Migration Guide</i> | Gives guidance for upgrading ChangeMan ZMF. |
| <i>OFM Getting Started Guide</i> | Explains how to install and use the Online Forms Manager (OFM) option of ChangeMan ZMF. |
| <i>SER10TY User's Guide</i> | Gives instructions for applying licenses to enable ChangeMan ZMF and its selectable options. |
| <i>User's Guide</i> | Describes how to use ChangeMan ZMF features and functions to manage changes to application components. |

| Manual | Description |
|--------------------------------------|---|
| <i>XML Services User's Guide</i> | Documents the most commonly used features of the XML Services application programming interface to ChangeMan ZMF. |
| <i>ZMF Web Services User's Guide</i> | Documents the Web Services application programming interface to ChangeMan ZMF. |

Using the Manuals

Use Adobe® Reader® to view ChangeMan ZMF PDF files. Download the Reader for free at get.adobe.com/reader/.

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 are delivered 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 Documentation tab on the Micro Focus SupportLine 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.

Using 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 |
|----------------|---|
| <i>italics</i> | 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. |

| Convention | Explanation |
|-------------------------------|--|
| <i>monospaced italics</i> | Indicates names that are placeholders for values you specify; for example, <i>filename</i> . |
| vertical rule | Separates menus and their associated commands. For example, select File Copy means to select Copy from the File menu. Also, indicates mutually exclusive choices in a command syntax line. |

Notes

Sterling Connect:Direct® is an IBM® point-to-point file transfer software product that can be used to transfer files between two ChangeMan ZMF instances. The original name of the product was Network Data Mover (NDM). The "NDM" mnemonic persists, embedded in Connect:Direct and ChangeMan ZMF component names, options, and JCL examples.

Chapter 1

Introduction

The ChangeMan ZMF rules-based environment for software configuration management provides processes based on best practices for managing application components.

Software change management can be expressed differently in different companies. ChangeMan ZMF architecture allows customers to modify details of the development lifecycle process. While user interfaces for ChangeMan ZMF have expanded beyond the host environment, many key functions are based on batch processing that can be customized to fit your requirements.

This design provides flexibility for you to quickly modify ChangeMan ZMF to fit your needs. However, there are some general recommendations that you should follow to protect the integrity of your ChangeMan ZMF components. These recommendations are detailed in the sections that follow.

| | |
|--|----|
| Preserving Vendor Versions of ChangeMan ZMF Components | 16 |
| Using ChangeMan ZMF To Manage ChangeMan ZMF Components | 16 |
| Nomenclature | 16 |

Preserving Vendor Versions of ChangeMan ZMF Components

Preserve the versions of components that are delivered in the ZMF installer. Do not edit components in the mainframe libraries unloaded from the installer. Allocate *custom* libraries to concatenate over vendor (delivered) libraries in the SERNET started procedure and other JCL that use ChangeMan ZMF libraries.

If you preserve the delivered version of components, you can return to the original version if modifications you make do not work as expected.

Using ChangeMan ZMF To Manage ChangeMan ZMF Components

We recommend that you use ChangeMan ZMF to manage ChangeMan ZMF components. This means that you create an application in production ChangeMan ZMF and create library types for ISPF skeletons, ISPF panels, ISPF messages, source code, load modules, and JCL. The baseline or production libraries for this application are concatenated under the ISPSLIB, ISPLLIB, ISPLMLIB, and STEPLIB of the SERNET started procedure.

To modify a ChangeMan ZMF component such as a skeleton, you create a change package, check out and edit the skeleton, then promote the package to populate a test skeleton library concatenated in the ISPSLIB ddname of a test ChangeMan ZMF instance. After testing the skeleton in a test ChangeMan ZMF environment, you audit and freeze the package, then obtain approvals that include management responsible for your change management software. When the package is installed, your production ChangeMan ZMF automatically starts using the new version of the skeleton.

For debugging purposes, users should copy their customized skeletons to a CUSTOM.SKELS library, and concatenate the CUSTOM.SKELS library ahead of the skeleton library distributed with the ChangeMan ZMF product.

Chapter 2 ISPF Skeletons provides details on customizing ChangeMan ZMF panels and variables. Chapter 3 User Exits describes exit functions, where they are invoked, and common uses. Extensive help is built into the ISPF environment, pressing the PF1 key once gets you some information, pressing the PF1 key a second time for ChangeMan ZMF Messages will get you more information. See the ChangeMan ZMF Messages Guide for further message details.

Nomenclature

Mainframe components of **SERNET** run as **started tasks** under z/OS. Each SERNET started task is assigned a unique one-character **subsystem ID**.

ChangeMan ZMF runs as an **application** under SERNET technology. It uses the subsystem ID assigned to the SERNET started task, but SERNET requires the subsystem ID even when there is no ChangeMan ZMF application.

One occurrence of SERNET is referred to as a SERNET **instance**. One occurrence of ChangeMan ZMF is referred to as a ChangeMan ZMF **instance**.

The ChangeMan ZMF programs that run under SERNET are called ChangeMan ZMF **server** programs. ChangeMan ZMF programs that run in the user address space, such as the ChangeMan ZMF ISPF interface, are referred to as ChangeMan ZMF **client** programs.

Chapter 2

ISPF Skeletons

This chapter tells you how to use the flexibility of standard IBM ISPF services to build your own change management processes that run within the secure environment of ChangeMan ZMF.

| | |
|---|----|
| Introduction | 20 |
| Skeleton File Tailoring in ChangeMan ZMF | 20 |
| CMN\$\$JBL - JOBLIB / STEPLIB | 31 |
| Setting Build Parameters | 33 |
| Transmit Selected Remote Promote Components | 36 |
| JES Node Names and Transmission Site Names | 36 |

Introduction

ChangeMan ZMF uses standard ISPF services to build batch job JCL from ISPF skeletons. This design provides extraordinary flexibility through standard IBM facilities. You can build your own batch processes inside ChangeMan ZMF for component builds and other processes while ChangeMan ZMF maintains the integrity of your software change management processes by securing development and production libraries and allowing only authorized access to its functions.

The *ChangeMan ZMF Installation Guide* describes skeletons that you must modify to bring up a test or demonstration ChangeMan ZMF instance and process a change package through the package life cycle. This chapter provides more details about customization in a complex ChangeMan ZMF environment.

A routine is available which allows the notification of Job Completion messages via email (only for z/OS 2.3 or greater). See the sample skeleton CMN\$\$ENT.

Skeleton File Tailoring in ChangeMan ZMF

File tailoring obtains variable values from ISPF variable pools populated by ChangeMan ZMF programs. It obtains ISPF skeletons from libraries concatenated under the ISPSLIB ddname in the SERNET started task JCL. ChangeMan ZMF file tailoring builds batch JCL to perform the following functions:

- Checkout in batch
- Compile, assemble, and link edit (build) procedures
- Promotion
- Audit
- Audit Auto Resolve
- Batch freeze
- Package distribution, installation, and baseline ripple
- Package backout
- Utility functions such as Component Compare, Print, and Copy (export)

Skeletons are delivered in the CMNZMF SKELS library in the ZMF installer.

Skeleton Naming Conventions

When ChangeMan ZMF was first released, there was a rigid naming convention for skeletons that conveyed their purpose and their position in the skeleton imbed hierarchy. As the product matured and the number of skeletons multiplied, the naming conventions lost their rigor.

However, where you see certain structures in a skeleton name, you can still infer information about the skeleton.

The table below explains the conventions in this sample skeleton name:

aaaiifff

where:

| | |
|-----|---|
| aaa | The first three characters of the delivered ChangeMan ZMF skeletons are CMN, which is an abbreviation for the product name. |
| ii | The following values usually carry the listed meaning: \$\$ Subordinate skeleton in an imbed hierarchy. IM IMS Option skeleton. IN Install skeleton, often for the IMS Option. JS Skeleton to insert a new JOB statement after 255 steps in generated JCL. PR Promotion skeleton, often for the IMS Option. RP Skeleton for promotion to a remote site. Zn Online Forms Manager skeleton. nn Installation job skeleton. |
| fff | Three-character acronym or abbreviation for the skeleton function. |

We recommend that you use an abbreviation for your company name as the first three characters of a skeleton name when you create a custom skeleton that is not a derivative of a skeleton that is delivered with ChangeMan ZMF.

Skeleton Variables

ISPF variables are used in ChangeMan ZMF skeleton logic to:

- Provide values for component names, data set names, parameters, subparameters, and other elements of JCL.
- Provide the conditions for file tailoring logic to include or exclude JCL statements.

Skeleton Variable Example

The following code fragment from a ChangeMan ZMF skeleton provides an example of both variable functions:

```
)SEL &DB2PCLL NE &Z
//STEPLIB DD DISP=SHR,DSN=&DB2PCLL
)ENDSEL &DB2PCLL NE &Z
```

In this example, if variable DB2PCLL is not a null value (blank), then the STEPLIB statement is included in the JCL generated by file tailoring.

If the STEPLIB statement is included in the generated JCL, the data set name of the library will be the value stored in variable DB2PCLL.

Where Variables Are Defined

ISPF variables are made available for file tailoring by these ChangeMan ZMF facilities:

- ChangeMan ZMF base product programs that set variable values based on conditions in the package master file, component master file, files under ChangeMan ZMF control, and values entered by users on standard ChangeMan ZMF panels.

- Package User Information variables entered by package creators if the Package User Information facility is turned on by the global administrator. See "[Package User Information](#)" on page 60.
- Stage User Option variables set by users on the custom ChangeMan ZMF User Option Panel built by the administrator or technician responsible for customizing ChangeMan ZMF. See "[Staging User Options](#)" on page 64.
- Skeletons CMN\$\$DSN, CMN\$\$VAR, CMN\$PARM, PRM\$aaaa, and VAR\$aaaa that you customize to set variables used in build processing.
- Release ID variables set by the global administrator. See "[Release ID Variables](#)" on page 71.
- Custom variables V01-V10 passed from ISPF panels to file tailoring for some ChangeMan ZMF batch jobs.

#VARLIST

Member #VARLIST in the CMNZMF SKELS library lists ISPF variables and variable tables defined in base ChangeMan ZMF programs.

- The variables and tables are grouped under the ChangeMan ZMF function that defines them.
- The variables in each table are listed.
- For each variable, the variable length and a short definition are provided.
- High level skeletons for each ChangeMan ZMF function are listed.

Skeleton Imbedding

ChangeMan ZMF uses the imbed facility of ISPF file tailoring to reduce redundancy. Common functions are coded in a skeleton. The common skeleton is then imbedded in other skeletons with the)IM control statement.

Imbedded skeletons can contain imbeds for other skeletons. ISPF file tailoring limits imbeds to 15 levels of imbedding, if you attempt further you will get an error with the skeleton name and record number that attempted to exceed that limit e.g. **Exceeds maximum)IM level of 15, CMN014 record-3**

Each ChangeMan ZMF skeleton begins with a JCL comment (except skeletons that generate JOB statements). This JCL comment contains the name of the skeleton. ISPF file tailoring passes JCL comments in skeletons directly to the output JCL. You can find the names of all of the ChangeMan ZMF skeletons used to generate a job by looking for JCL comments that look like this:

```
/**)IM CMNxxxxx
```

The sequence of these JCL comments shows the sequence of skeletons processed by ISPF file tailoring to generate a job. The sequence may mean that a skeleton is imbedded in a skeleton named previously in a JCL comment in the job.

See [Appendix B, "Skeleton Hierarchy"](#) on page 171 for tables that show the hierarchy of imbedded ChangeMan ZMF skeletons. The appendix also provides instructions for analyzing skeleton imbed hierarchies, which you can use to analyze your customized skeleton structures.

Skeleton Maintenance Facility

ChangeMan ZMF includes a skeleton maintenance facility that the global administrator can use to:

- Customize skeletons
- Check skeleton syntax
- Create and maintain application-level variables called Release ID variables for use in file tailoring for batch job JCL.

Accessing Skeleton Maintenance

To display the **Skeleton Maintenance Options** menu, use one of these two methods.

- Access the **Skeleton Maintenance Options** panel directly by typing **=A.G.S** and pressing **Enter**,

or

- Follow these steps to access the **Skeleton Maintenance** panel using ChangeMan ZMF menus:
 - a On the **Primary Option Menu**, select **option A Admin**.
 - b On the **Administration Options menu**, select **option G Global**.
 - c On the **Global Administration Options menu**, select option **S Skeleton**.

The **Skeleton Maintenance Option** menu (CMN3DSKL) is displayed.

```

CMN3DSKL                Skeleton Maintenance Options
Option ==> _____
M Maintain              Maintain skeleton release variables
A Assist                File tailoring assistance of skeleton procedures
  
```

Options on the **Skeleton Maintenance Options** menu include:

- **M Maintain** to take you to the **Maintain Release ID Variables** panel where you create and update release ID variables. See ["Developing Skeletons With File Tailoring Assistance" on page 23](#).
- **A Assist** to take you to the **File Tailoring Assistance** panel where you can edit skeletons and test skeleton syntax. See ["Release ID Variables" on page 71](#).



CAUTION! Never select **A Assist** on a production ChangeMan ZMF instance.

When you select **A Assist**, ISPF skeleton libraries are immediately enqueued, and skeleton file tailoring cannot be executed.

Developing Skeletons With File Tailoring Assistance

We recommend that you use ChangeMan ZMF to manage ChangeMan ZMF components. See ["Using ChangeMan ZMF To Manage ChangeMan ZMF Components" on page 16](#). However, early in the initial implementation of ChangeMan ZMF, there may be justification for using the File Tailoring Assistance facility in ChangeMan ZMF global administration to modify skeletons.

File Tailoring Assistance automatically obtains a skeleton you want to edit or validate from the first occurrence of the skeleton in the ISPSLIB concatenation of the SERNET started task. Editing is performed in an ISPF edit session running under ChangeMan ZMF. When edit changes are saved, ChangeMan ZMF saves the customized skeleton back into the top library in the ISPSLIB concatenation. The top library in the ISPSLIB concatenation should be your custom skeleton library.

Advantages of Using File Tailoring Assistance

The advantages of using File Tailoring Assistance include:

- Rapid editing of skeletons.
- Automatic preservation of vendor versions of skeletons (if you properly allocate a custom skeleton library and concatenate it at the top of the ISPSLIB libraries in your started task JCL).
- Syntax checking of complex skeleton logic without creating application, component, and user conditions that will create variable values to exercise that logic.
- Rapid testing of skeletons from the same ChangeMan ZMF instance where they were edited.

Disadvantages of File Using Tailoring Assistance

The disadvantages of using File Tailoring Assistance include:

- No skeleton versions are preserved between the original vendor version and the current custom version that is running the ChangeMan ZMF instance.
- There is no audit trail of skeleton changes.
- There is no guarantee that variable values assigned in File Tailoring Assistance are available in the ChangeMan ZMF function where the file tailoring is actually performed.
- All libraries in the ISPSLIB concatenation are enqueued while File Tailoring Assistance facility is in use. Execution of batch functions on this ChangeMan ZMF instance are blocked.



CAUTION! For this reason, never use File Tailoring Assistance on a production ChangeMan ZMF instance.

- Skeleton developers must be granted update access to the custom skeleton library. This is not acceptable if the library is used to run production ChangeMan ZMF instances.
- When skeleton changes are saved, the changes are effective immediately. There is no promotion facility to test the skeleton outside of File Tailoring Assistance.

Recommended Use of File Tailoring Assistance

The disadvantages listed above make the use of File Tailoring Assistance inappropriate in a ChangeMan ZMF instance that manages production components.

File Tailoring Assistance may be used in an initial implementation of ZMF:

- 1 Bring up the new ChangeMan ZMF instance

- 2 Use File Tailoring Assistance to perform initial skeleton modifications to get ESSENTIAL batch jobs to run successfully.
- 3 Create a ChangeMan ZMF application, and continue development of custom components under ChangeMan ZMF supervision.

Editing Skeletons in File Tailoring Assistance

- 1 Display the **Skeleton Maintenance Options** menu using one of these two methods.
 - Access the **Skeleton Maintenance Options** panel directly by typing **=A.G.S.A** and pressing **Enter**,

or

 - On the **Skeleton Maintenance Option** menu, select option **A Assist**. The **File Tailoring Assistance** panel (CMN3DSA0) is displayed.

```

CMN3DSA0                                File Tailoring Assistance
Option ==> _____

      blank Display skeleton list          E Edit skeleton
          S Select for file tailoring      V View skeleton

Skeleton name . . . CMN11 (Blank for list; required for options E,V,S)
Release id . . . .        (Blank for list)
Application . . . . ACTP (Blank for list)

```

- 2 On the **File Tailoring Assistance** panel:
 - a Type **E Edit** in the **Option** line.
 - b Type the name of a skeleton in the **Skeleton Name** field.
 - c Press **Enter**.

The skeleton you named is opened in an ISPF edit session.

```

ISREDD2  CMNTP.CMN810.C6.SKELS(CMN11) - 00.00          Columns 00001 00072
Command ==> _____ Scroll ==> CSR
***** ***** Top of Data *****
000001 )IM CMN$$SJM
000002 //*)IM CMN11
000003 //*
000004 //*  JOB TO INSERT &PKGNAME INFORMATION IN PACKAGE MASTER AT &RMTSITE
000005 //*
000006 )CM
000007 )CM  UPDATE DDNAME CMN11ENQ WITH A VALID DATASET NAME BEFORE USING;
000008 )CM  THIS DATASET ELIMINATES CONFLICTS WITH RECORDS BEING UPDATED IN
000009 )CM  THE PACKAGE MASTER FROM THE .PACKAGE DATASET
000010 )CM
000011 )SEL &RSTTTYP EQ FTP
000012 )CM
000013 )CM  RECEIVE PACKAGE DATASET
000014 )CM
000015 )SET RCVDSN  = &PRSPKG
000016 )SET STEPID  = PACK
000017 )IM CMN$$RCV
000018 )CM
000019 )CM  RECEIVE INSTALLATION "X" DATASET
000020 )CM
000021 )SET RCVDSN  = &PRSNOD
000022 )SET STEPID  = DOTX
000023 )IM CMN$$RCV
000024 )CM
000025 )CM  RECEIVE ALL THE STAGING DATASETS
000026 )CM
000027 )CM  SPECIFY THE REQUIRED PARAMETERS FOR THE CMNSUBIR SKELETON:
000028 )CM
000029 )SET SUBDOT   = &RMLBTBL
000030 )SET SUBPARJS = 230
000031 )SET SUBNR1SK = &Z
000032 )SET SUBNR1JS = 0
000033 )SET SUBREPSK = CMN$$F08
000034 )SET SUBREPJS = 4
000035 )SET SUBNR2SK = &Z
000036 )SET SUBNR2JS = 0
000037 )SET SUBERRSK = CMN11ERR

```

- 3 Use standard ISPF edit commands to change the skeleton.
- 4 Press **PF3** or type **END** and press **Enter** to save the edit changes and return to the **File Tailoring Assistance** panel.
- 5 If File Tailoring assistance found the skeleton in the library at the top of the ISPSLIB library concatenation in the started task JCL, the updated member is saved back to that library. If the skeleton was found in another library lower in the concatenation, it is saved to the top library.



NOTE Editing in File Tailoring Assistance is not the same as editing a ChangeMan ZMF package component. In File Tailoring Assistance, you are editing directly in the skeleton library, and there is no compare listing or commit process when you press PF3. Your changes are simply saved directly into the live data set.

Syntax Checking in File Tailoring Assistance

With File Tailoring Assistance, you can check the syntax of skeletons you have modified. A simulated file tailoring session resolves variable substitutions and presents the resulting file on the screen for review.



NOTE File Tailoring Assistance does not resolve variables with values provided by ChangeMan ZMF panels and functional programs. Likewise, ISPF tables are not built. You can provide values for variables using the Release ID Variable facility described in the next section.

- 1 Display the **Skeleton Maintenance Options** menu using one of these two methods.
 - Access the **Skeleton Maintenance Options** panel directly by typing **=A.G.S.A** and pressing **Enter**.
 - or
 - On the **Skeleton Maintenance Option** menu, select option **A Assist**. The **File Tailoring Assistance** panel (CMN3DSA0) is displayed.

```

CMN3DSA0                                File Tailoring Assistance
Option ==> S _____

      blank Display skeleton list          E Edit skeleton
      S Select for file tailoring          V View skeleton

Skeleton name . . . CMN11 (Blank for list; required for options E,V,S)
Release id . . . . . (Blank for list)
Application . . . . . ACTP (Blank for list)
  
```

- 2 On the **File Tailoring Assistance** panel:
 - a In the **Application** field, type the name of the application that contains the skeleton you want to test.
 - b Type the name of a skeleton in the **Skeleton Name** field and press **Enter**, or leave the field blank and press **Enter** to display the **Skeleton Member List** panel, then type **S** in the **line command** for a skeleton and press **Enter**. You may have to select a Release ID first from the **Release Id List** panel (CMN3DSA2)

```

CMN3DSA2                                Release Id List                                Row 1 to 1 of 1
Command ==> _____ Scroll ==> CSR_____

      Id      Changed      User      Description
S V6R1 2008/07/09 13:35 USER25
***** Bottom of data *****
  
```

```

ISREDDE2      CMNTP.A008A.#CE259A5.#75A7E25                Columns 00001 00072
Command ==>          Scroll ==> CSR
***** ***** Top of Data *****
000001 //S8CPRMA JOB (SM-1IKF-SM),'CMN',
000002 //          CLASS=A,
000003 //          NOTIFY=USER25,
000004 //          MSGCLASS=X
000005 //*
000006 /** THE ABOVE JOB CARDS CAME FROM THE IMBED OF SKEL CMN$$JCD
000007 /**)IM CMN$$JCD
000008 /**
000009 /** JOB REQUESTED BY USER25 ON 2014/12/02 AT 21:59
000010 /**
000011 /**)IM CMN$$DSN
000012 /**)IM CMN$$JBL
000013 //JOBLIB DD DISP=SHR,DSN=CMNTP.CMN810.C6.LOAD
000014 //          DD DISP=SHR,DSN=CMNTP.SER810.C6.LOAD
000015 //          DD DISP=SHR,DSN=CMNTP.CMN810.LOAD
000016 //          DD DISP=SHR,DSN=CMNTP.SER810.LOAD
000017 #%
000018 /**)IM CMN$$SJM
000019 /**
000020 /** PACKAGE FROZEN BY USER015 ON 2014/12/02 AT 21:59
000021 /**
000022 /**)IM CMN$$DSN
000023 /**)IM CMN$$JBL
000024 //JOBLIB DD DISP=SHR,DSN=CMNTP.CMN810.C6.LOAD
000025 //          DD DISP=SHR,DSN=CMNTP.SER810.C6.LOAD
000026 //          DD DISP=SHR,DSN=CMNTP.CMN810.LOAD
000027 //          DD DISP=SHR,DSN=CMNTP.SER810.LOAD
000028 /**)IM CMN11
000029 /**
000030 /** JOB TO INSERT ACTP000008 INFORMATION IN PACKAGE MASTER AT
000031 /**
000032 /**)IM CMN00INS
000033 //CMN00 EXEC PGM=CMNBATCH, *** Access ChangeMan ZMF started task
000034 //          PARM='SUBSYS=6',
000035 //          COND=(4,LT)
000036 /**)IM CMN$$SPR
000037 //SER#PARM DD DISP=SHR,DSN=CMNTP.SER810.C6.TCPIPORT

```

- The JCL created from the file tailored skeleton is displayed, at least up to the point where a syntax error was detected or file tailoring assistance tried to resolve a variable or open a table that does not exist. Error messages from file tailoring are displayed at the top of the JCL.



NOTE The record numbers referenced in error messages are the skeleton record numbers, not records in the output JCL.

Debugging Skeletons in Started Task Procedures

Starting in ChangeMan ZMF 5.1, ISPF file-tailoring is migrated from your TSO address space to file tailoring address spaces that are initiated by the SERNET started task that runs ZMF. You have no access to these separate address spaces, so you cannot directly test file tailoring for custom skeletons.

CLIST CMNDBGAS executes ChangeMan ZMF programs in your address space. You can run this CLIST in tools like ISPF Dialog Test to debug custom skeletons.

File Tailoring Procedure Names

In releases prior to ChangeMan ZMF 5.6, ISPF file-tailoring is performed by a single started procedure named CMNxADSP, where x is the subsystem ID of the started task under which ChangeMan ZMF runs.

Since ChangeMan ZMF version 5.6, ChangeMan ZMF administrators can specify up to four different procedures that perform the following file-tailoring functions:

- Installation JCL builds.
- Batch component builds.
- Promotion JCL builds.
- All other file-tailoring functions.

The ChangeMan ZMF global administrator identifies these procedures on the Global Parameters, Part 2 of 8 [CMNGGP02] panel. Refer to the *ChangeMan ZMF Administrator's Guide* for a description of this panel.

If there is a failure in the execution of a file tailoring started task, it sends a message to the console log. You can use the JobID associated with the console log message to find the file tailoring server output in SDSF (or another sysout viewing tool) that contains information about the cause of the failure.

Considerations

- You must have global administrator authority to run the started task procedure address space programs in your TSO address space.
- CLIST CMNDBGAS uses control information that is generated by a file tailoring started procedure when the TRACE facility is enabled in the ZMF started task. When TRACE is on in the ZMF instance, the spawned file tailoring started task writes the control information to dynamically allocated sysout DDname RQST.
- CMNDBGAS reads the control information from a sequential file. Before executing CMNDBGAS, you must allocate the sequential file and copy the control information from the RQST sysout data set into the new file.
- The method you use to copy the RQST sysout records to a cataloged data set depends on the facilities and requirements at your site. The procedure described in this section assumes that you use your sysout viewing tool to copy that information into a cataloged data set.
- Execute the procedure described in this section using your test ChangeMan ZMF system. The TRACE facility generates a significant volume of output, which could be harmful in a ChangeMan ZMF instance being used by many developers.

Set Up CLIST CMNDBGAS

- 1 Copy member CMNDBGAS from the vendor CLIST library to your custom CLIST library.
- 2 Code an appropriate SYSOUT class for DDnames SERPRINT and RQST.
- 3 Replace the ALLOC statement for ddname SER#PARM with the same statement in your log-on CLIST for ChangeMan ZMF.
- 4 Replace the LIBDEF statements for ISPLLIB, ISPMLIB, and ISPLLIB with the LIBDEFs in your ZMF logon CLIST.

- 5 Replace the library concatenation in the LIBDEF ISPSLIB statement with the concatenation in your ZMF started procedure.
- 6 To ensure a clean ISPF variable environment, ensure that CLIST CMNDBGAS specifies a NEWAPPL() application id that does not contain ChangeMan ZMF variables in the profile pool member in ISPPROF.

Run CLIST CMNDBGAS

The instructions in this section assume that you use SDSF to browse spool output and are authorized to issue the commands shown. Substitute appropriate steps and commands if you do not have the required authorization or use another sysout viewing tool.

- 1 Turn on the TRACE facility with the following modify command:

```
F server,TRACE,ON,CMN,CLASS=1
```

where *server* is the jobname of the SERNET started task that runs ZMF.

- 2 Execute your logon CLIST to connect to ChangeMan ZMF through the ISPF interface, and execute the ZMF function that will file tailor the skeleton you want to test. The purpose of this step is to capture the control information required to run this function with CLIST CMNDBGAS.

- 3 Turn off the TRACE facility with the following modify command:

```
F server,TRACE,OFF,CMN,CLASS=1
```

where *server* is the jobname of the SERNET started task that runs ZMF.

- 4 Allocate a small sequential data set *somnode.CMNADSP.REQUEST* with DCB:

```
RECFM=FB  
LRECL=1000
```

This file will only contain a few records.

- 5 Enter SDSF and set PREFIX to the name of the file tailoring started task. Display the SDSF queue appropriate for the SYSOUT class coded in the file tailoring started procedure.
- 6 Type **?** in the line command for the file tailoring started task sysout and press **Enter**.
- 7 Type **SE** in the line command for the RQST ddname and press **Enter** to display the sysout data in edit mode.
- 8 Copy the contents of the RQST sysout data set into *somnode.CMNADSP.REQUEST* that you allocated previously:
 - a In the line command for the first record, type **C9999**.
 - b On the **Command** line, type **REPLACE** and press **Enter**.
 - c On the **Name** line of the **Edit/View Replace** panel, type DSN *somnode.CMNADSP.REQUEST* and press **Enter**.
 - d Press **Enter** on the **EDIT - Confirm Replace** panel, ignoring the difference in the record format.
 - e Exit the RQST sysout data set.

- 9 Invoke CLIST CMNDBGAS from within a debugging environment, such as ISPF dialog test (Note that this CLIST is distributed in the supplied CLIST library for ChangeMan ZMF and will need to be updated and placed in a custom library before you can use it).
- 10 When prompted, enter the data set name *somnode.CMNADSP.REQUEST*.

ISPF Table CMNTBN

Programs running in a started task procedure address space start with a clean ISPF environment. To ensure that no ChangeMan ISPF tables are left open in a user's address space, a TBEND is issued for each ISPF table that was created during the session.

These TBENDs occur when you exit from a client ChangeMan ZMF session in an ISPF environment. ChangeMan knows which tables to TBEND by maintaining another ISPF table called CMNTBN. CMNTBN contains a list of all ISPF tables created during the ChangeMan session.

Table CMNTBN may help you when you are debugging under ISPF dialog test.

Error Codes

- User abend 6 is issued if the user cannot connect to the ChangeMan ZMF instance.
- User abend 10 is issued if the saved request block data set cannot be opened.

CMN\$\$JBL - JOBLIB / STEPLIB

We recommend against including ChangeMan ZMF load libraries in the LINKLIST. ChangeMan ZMF includes skeleton CMN\$\$JBL, which provides a standard JOBLIB concatenation for batch jobs submitted from the started task.

Skeleton CMN\$\$JBL is also used to provide ZMF libraries in a STEPLIB for plan lookup program CMNDB2PL in the DB2 Option. In file tailoring, the DD name defaults to JOBLIB unless variable &JOBLBDD is set to STEPLIB or some other value.

Skeleton CMN\$\$JBL is included by these skeletons.

| Skeleton | Description | DD Name |
|------------|---|---------|
| CMN\$\$BRQ | Routine to connect to a remote task for backouts and reverts | STEPLIB |
| CMN\$\$D2J | JOB statement for DB2 bind jobs | JOBLIB |
| CMN\$\$JCD | JOB statement for user-initiated jobs | JOBLIB |
| CMN\$\$JNM | JOB statement for baseline ripple and other installation jobs at the development instance | JOBLIB |
| CMN\$\$NTF | Routine to invoke batch approval notification | JOBLIB |
| CMN\$\$PRB | Routine for DB2 binds at promotion or demotion | STEPLIB |
| CMN\$\$RAL | Routine for release audit reporting tables update | STEPLIB |
| CMN\$\$RAP | Routine for generate release audit reports | STEPLIB |
| CMN\$\$RAU | Routine for audit area | STEPLIB |

| Skeleton | Description | DD Name |
|------------|---|---------|
| CMN\$\$RPB | Routine for DB2 binds at remote promotion or demotion | STEPLIB |
| CMN\$\$RPJ | JOB statement for remote promotion or demotion | JOBLIB |
| CMN\$\$SJM | JOB statement for remote site installation or backout | JOBLIB |
| CMN\$\$SPB | Sample routine for promotion DB2 binds using local shadow libraries | STEPLIB |
| CMN21 | Routine for DB2 binds for production library installation | STEPLIB |
| CMN32 | Routine for DB2 binds for baseline ripple | STEPLIB |
| CMN49 | Routine for DB2 binds for production library backout | STEPLIB |
| CMN56 | Routine for DB2 binds for reverse baseline ripple | STEPLIB |

The delivered skeleton contains selection logic to concatenate test ZMF load libraries in front of production ZMF libraries for a test instance of ChangeMan ZMF. The actual data set names are in variables that are defined in the skeleton CMN\$\$DSN.

If you license the ChangeMan ZMF DB2 Option, do not include a DB2 system load library in skeleton CMN\$\$JBL. Skeletons that create JCL for DB2 bind jobs obtain the DB2 library name from variable &STSL0D using logic like this (example taken from CMN\$\$PRB):

```
//DB2PL EXEC PGM=CMNDB2PL, *** DETERMINE DB2 BIND REQUIREMENTS
//          REGION=0M,
//          COND=(4,LT)
)SET JOBLBDD = STEPLIB
)IM CMN$$JBL
)SEL &STSL0D NE &Z
)SET DB2DSNLD = &STSL0D
)IM CMN$$D2X
//          DD DISP=SHR,DSN=&DB2DSNLX
//          DD DISP=SHR,DSN=&STSL0D
)ENDSEL &STSL0D NE &Z
```

Variable &STSL0D contains the library name entered in the DB2 System Load Library field on the DB2 Physical Subsystems Part 1 of 2 (CMNGD2S0) panel in DB2 global administration.

Customization Tasks:

- 1 Copy skeleton CMN\$\$DSN from your vendor CMNZMF SKELS library to your custom SKELS library.
- 2 In the section that defines the JOBLIB, code the appropriate values to use delivered and CUSTOM load libraries for CMNZMF LOAD and SERCOMC LOAD libraries. This is as delivered (note the first half are for a test instance of ChangeMan ZMF, second half are for a production instance of ChangeMan ZMF):

```
)CM THIS DEFINES THE VARIOUS LOAD LIBRARIES USED TO BUILD THE JOBLIB
)CM CONCATENATION.
)CM (CMN$$JBL)
)CM
)SET ZMFCTST = somnode.CMNZMFt.CUSTOM.LOAD
)SET SERCTST = somnode.SERCOMCt.CUSTOM.LOAD
)SET ZMFTST = somnode.CMNZMFt.LOAD
)SET SERVST = somnode.SERCOMCt.LOAD
)SET ZMFCPRD = somnode.CMNZMF.CUSTOM.LOAD
)SET SERCPRD = somnode.SERCOMC.CUSTOM.LOAD
```



```
)SET ZMFVPRD = somnode.CMNZMF.LOAD
)SET SERVPRD = somnode.SERCOMC.LOAD
```

Note that in the distributed CMN\$\$DSN, the commented name in brackets tells you the name of the relevant skeleton(s) using these variables, in this example it is CMN\$\$JBL.

- 3 If you have a ChangeMan ZMF test instance, there is logic in CMN\$\$JBL based on subsystem ID so that your test libraries are concatenated in front of the libraries you use to run your production ChangeMan ZMF instance. You will also need to copy and edit the CMN\$\$JBL skeleton to use the correct subsystem ID for your test instance.

Setting Build Parameters

Build process parameters are provided to skeleton file tailoring from the following sources:

- 1 Skeleton CMN\$\$VAR - Initializes build parameter ISPF variables to blank.
- 2 Skeleton CMN\$PARM - Assigns system-wide values to build parameter variables.
- 3 Optional skeleton PRM\$aaaa (where aaaa is an application mnemonic) - Overlays system-wide default values with application specific values.
- 4 Compile parm and link edit parm fields set in the user interface and stored in component history.

Build Parameter ISPF Variables

These are build parameter variables initialized to a null value in CMN\$\$VAR, then set in "global" skeleton CMN\$PARM, and used in build process skeletons.

| Variable Name | Description |
|---------------|----------------------------------|
| &DB2PPRM1 | DB2 Precompile Parameters Part 1 |
| &DB2PPRM2 | DB2 Precompile Parameters Part 2 |
| &DB2PPRM3 | DB2 Precompile Parameters Part 3 |
| &CICSPRM1 | CICS Translate Parameters Part 1 |
| &CICSPRM2 | CICS Translate Parameters Part 2 |
| &CICSPRM3 | CICS Translate Parameters Part 3 |
| &MAPDPRM1 | BMS MAP DSECT Parameters Part 1 |
| &MAPDPRM2 | BMS MAP DSECT Parameters Part 2 |
| &MAPDPRM3 | BMS MAP DSECT Parameters Part 3 |
| &COMPPRM1 | Compile Parameters Part 1 |
| &COMPPRM2 | Compile Parameters Part 2 |
| &COMPPRM3 | Compile Parameters Part 3 |
| &PLNKPRM1 | Prelink Parameters Part 1 |

| Variable Name | Description |
|---------------|---------------------------|
| &PLNKPRM2 | Prelink Parameters Part 2 |
| &PLNKPRM3 | Prelink Parameters Part 3 |

These are the compile parameter and link edit parameter variables used in build process skeletons that are set in the user interface or retrieved from component history.

| Variable Name | Description |
|---------------|---------------------------|
| &COMPOPT1 | Compile Parameters Part 1 |
| &COMPOPT2 | Compile Parameters Part 2 |
| &COMPOPT3 | Compile Parameters Part 3 |
| &COMPOPT4 | Compile Parameters Part 4 |
| &COMPOPT5 | Compile Parameters Part 5 |
| &LINKPRM1 | Link Parameters Part 1 |
| &LINKPRM2 | Link Parameters Part 2 |
| &LINKPRM3 | Link Parameters Part 3 |

Build Parameter Skeleton Architecture

Skeleton CMN\$\$VAR is imbedded in every build procedure main skeleton. Skeletons CMN\$PARM and optional skeleton PRM\$aaaa are imbedded in CMN\$\$VAR. Variables &COMPOPTx and &LINKOPT are put into the ISPF variable pool by ChangeMan ZMF build process programs.

This code fragment from skeleton CMN\$\$VAR shows initialization of build parameter variables, the imbed of skeleton CMN\$PARM, and the imbed of optional skeleton PRM\$&PROJECT. An)IM command in a skeleton will be ignored if the OPT parameter is given and the skeleton is not found.

```

)SET COMPPRM1 = &Z
)SET COMPPRM2 = &Z
)SET COMPPRM3 = &Z
. . .
)CM
)CM SET GLOBAL (I.E. SHOP STANDARD) COMPILE, LINK, ETC. PARAMETERS.
)CM
)IM CMN$PARM
. . .
)CM
)CM PERFORM IMBED OF APPLICATION-SPECIFIC PARMS FOUND IN A "PRM$XXXX"
)CM SKELETON MEMBER, IF THIS MEMBER EXISTS.
)CM
)IM PRM$&PROJECT OPT

```

These code fragments from skeletons CMN\$\$CO2 and CMN\$\$LNK (named in bold on IM comment) show the build parm variables in the PARM statements for a COBOL2 compile step and a link edit step.

```

/**)IM CMN$$CO2
)CM
)CM ROUTINE TO COMPILE COBOL2 SOURCE CODE
)CM
. . .

```

```

//COBOL2&C#N EXEC PGM=IGYCRCTL, *** COMPILE COMPONENT &CMPNAME
//          COND=(4,LT),
//          PARM=('&COMPPRM1',
)SEL &COMPPRM2 NE &Z
//          '&COMPPRM2',
)ENDSEL &COMPPRM2 NE &Z
)SEL &COMPPRM3 NE &Z
//          '&COMPPRM3',
)ENDSEL &COMPPRM3 NE &Z
//          '&COMPOPT')
. . .

/**)IM CMN$$LNK
)CM
)CM ROUTINE TO LINK-EDIT A PROGRAM
)CM
. . .
//LINK&L#N!EXEC!PGM=IEWL, *** LINK-EDIT COMPONENT &CMPNAME
//          COND=(&CC$SUCC,LT),
//          PARM=('&LINKPRM1',
)SEL &LINKPRM2 NE &Z
//          '&LINKPRM2',
)ENDSEL &LINKPRM2 NE &Z
)SEL &LINKPRM3 NE &Z
//          '&LINKPRM3',
)ENDSEL &LINKPRM3 NE &Z
)SEL &TLODLIKE EQ N
//          'NCAL',
)ENDSEL &TLODLIKE EQ N
//          '&LINKOPT')
. . .

```

Customization Steps

Execute these steps to set values for build parameter variables.

- 1 Copy skeleton CMN\$PARG from your vendor CMNZMF SKELS library to your custom SKELS library.
- 2 Assign system-wide values to build parameter variables according to language, or code your own selection logic.
- 3 If necessary, create an optional application skeleton in your CUSTOM SKELS library.
 - a Copy skeleton PRM\$CMAN from your vendor CMNZMF SKELS library to your custom SKELS library.
 - b Rename the skeleton PRM\$aaaa, where aaaa is an application mnemonic.
 - c Assign application-level overrides to build parameter variables, only where they are different than system-wide values in skeleton CMN\$PARG.



NOTES Sample skeleton VAR\$CMAN can be renamed VAR\$aaaa, where aaaa is an application mnemonic, to overlay system-wide default values set in CMN\$\$VAR with application specific values.

Transmit Selected Remote Promote Components

With the promotion skeletons that are delivered with ChangeMan ZMF, when you selectively promote components to a remote site, all components in the staging library are transmitted to the remote site. Only the components you selected for promotion are copied from the transmitted library to the promotion library.

Skeleton code is available to create remote promote JCL that sends only the components selected for promotion to the remote site.

This facility was created for the ERO Option where promotion originates with release area libraries, which are typically much larger than package staging libraries. However, the function can be used to enhance performance of all remote promotion jobs in ChangeMan ZMF, not just ERO area promotion.

If you want to enable the skeleton code that creates more efficient remote promotion jobs, follow the instructions in comments at the top of these skeletons.

CMN\$RPM

CMNIMRPM

CMNRPMDL

If you want the more efficient remote promotion JCL to be created for all remote promotion jobs in ChangeMan ZMF, not just for ERO area promotion, follow the instructions in the comments at the top of this skeleton.

CMN\$PMT

JES Node Names and Transmission Site Names

Prior to ChangeMan ZMF 6.1 the values entered in the **Logical Unit/System Name** field in Global Administration Parameters and Global Site definitions were used for two purposes:

- JES node name that specifies where a ChangeMan ZMF batch job is run for promotion or install.
- Transmission site names that specify where install JCL, staging libraries, and package master records are transmitted from and to when they are distributed for installation.

Customers have noted that JES node names and transmission site names are not necessarily the same for a ChangeMan ZMF instance.

Since ChangeMan ZMF 6.1 this single field has been expanded and is now two fields: **Site node name** and **Logical unit/system name** on the **Global Parameters - Part 1 of 8** panel (CMNGGP01).

```

CMNGGP01                Global Parameters - Part 1 of 8
Command ===> _____

Subsystem: 6

ChangeMan ZMF environment . . . . DP          (A/D/DP/P)
Job entry system . . . . . JES2          (JES2 or JES3)
Site node name . . . . . SERT6
Logical unit/system name . . . . BUCKS
Default unit name . . . . . SYSDA
Default volume serial . . . . .
Default non-vio unit name . . . . SYSDA
ChangeMan ZMF security resource . $CMNTP
Default job scheduler . . . . . MANUAL    (CMN, Manual, Other)
Scheduler interval (CMN) . . . . 010     (Minutes)

Enter "/" to select option
/ Allow CMN scheduler
/ Allow Manual scheduler
/ Allow Other scheduler

```

- **Site node name** - If this is a DP, P or D site, the 'Site Name' (specified in Global Administration 'SITE') where packages will be installed. If an A site, enter an easily recognizable name (city, department, etc.).
- **Logical unit/system name** - If your data transmission vehicle is Connect:Direct® or BDT, enter the logical unitname (the name that Connect:Direct or BDT uses to identify this system). If IEBCOPY, enter the system name (e.g. 'SYSA').

Chapter 3

User Exits

ChangeMan ZMF exits are programs that are called by base product programs. Exits permit you to alter the processing of base product programs depending on the information passed to the exit.

| | |
|--|----|
| Introduction | 40 |
| User Exit Source | 40 |
| User Exit Interface Data | 40 |
| Customizing Exits | 41 |
| Calling XML Services from User Exits | 44 |
| Exit Descriptions | 45 |

Introduction

ZMF 8.1 provides central high-level language exit services that can be called by any client that can connect to ZMF. You can code the exits in any Language Environment (LE) compliant language as well as REXX. The same customer-supplied exit code will be executed regardless of which client is being used. For details of the new HLL Exits, please see the *ChangeMan ZMF 8.1 High Level Language Functional Exits Getting Started Guide*.

Please also note HLL exit points have no relation to, and do not replace, any existing ZMF assembler exit points described in this chapter.

Using the assembler exits, you can:

- Change defaults and settings for ChangeMan ZMF interaction with the operating system
- Alter what ChangeMan ZMF will do under specific component, package, or life cycle conditions
- Change who has authority to initiate specific ChangeMan ZMF tasks
- Change when authorized users can perform certain tasks

Common reasons for using exits include:

- Adapt ChangeMan ZMF to your data center standards and environment
- Enforce your company's standards, processes, and procedures for software change management
- Implement custom processes to manage unique component build processes
- Add rule-based flexibility to standard ChangeMan ZMF processing

User Exit Source

Source for user exits is delivered in the CMNZMF ASMSRC library in the ChangeMan ZMF installer.

When you modify exit source, preserve the delivered version in the delivered source library. See ["Preserving Vendor Versions of ChangeMan ZMF Components" on page 16](#).

User Exit Interface Data

The data passed between calling programs and exits is defined in copybooks in the CMNZMF ASMCPY library. This library is delivered in the ZMF installer.

Do not modify copybooks used to call exit programs. Even if you code the exit to handle the altered copybook, you cannot change the control block used to call the exit from the base product program.

Most exits exchange data with the calling program through two copybooks:

- CMNEXITS - This copybooks is common to nearly all ChangeMan ZMF exits.

- CMNEX nnn - The name of the copybook matches the name of the exit program in which it is included.

Not all fields in copybooks CMNEXITS and CMNEX nnn are populated by the calling program. See the comments at the top of the exit program source for a list of fields in these copybooks that are available to the exit you are customizing.

No Access to TCA

Starting with ChangeMan ZMF 5.5, you cannot access the TCA in user exits. The TCA register no longer points to the TCA when an exit is called. Any attempt to reference the TCA in a user exit results in an S0C4 abend.

Customizing Exits

ChangeMan ZMF exit programs are delivered inactive, except for exit CMNEXINS. Follow these steps to customize and activate an exit to modify ChangeMan ZMF behavior.

Find the Exit You Want

See ["Exit Descriptions" on page 45](#) to find an exit that targets the function you want to modify. Review the examples in the exit description to see how the exit can be used to alter the target function.

Read the comments at the top of the exit source member for more information about the exit. Review the sample code in the exit to see if the exit can perform the function you want.

Modify Exit Source

Follow these steps to modify the exit source to perform the function you want.

- 1 Check out the exit source in your ChangeMan ZMF application. If you do not have a ChangeMan ZMF application, copy the source from the delivered ASMSRC library to your custom ASMSRC library.
- 2 Edit the source to enable the exit. Comment out the following code to cause the exit program to be loaded at ChangeMan ZMF started task initialization:

```
*****
*
* Comment (or delete) the following 2 lines to activate this exit.
*
CMNEX014 CSECT
          DC      Y(2046)          inactive module
*****
```

- 3 Modify the exit source. Many exits are delivered with examples that can be used as they are delivered, or the sample code can be modified to perform the function you desire.

- a Some exits contain tables with entries that can be changed and extended to serve the purpose you need. Examine the end of the table to see how the program detects the end of table data, and make adjustments if necessary.
 - b If necessary, set the exit return code. If sample code is provided, it usually sets the return code to the appropriate value to communicate the exit result to the calling program. Valid values for return codes are described in comments at the top of source members.
- 4 Use coding techniques that will make it easy to see and understand your modifications when they must be updated or applied to the next release of ChangeMan ZMF.

If you comment out delivered code or overtype it, compare tools will often show the custom code as interleaved sets of inserted and deleted lines that are hard to understand. Use one of the following techniques to make code compare results easier to read:

- Code custom table entries as a separate copybook component, and insert a `COPY` statement into the exit code in place of the delivered sample table. To update the table, change the copylib member and reassemble the exit.
- Use an `AGO` statement to exclude delivered code, then place your custom code beneath the excluded code.

```

****
* Valid work request number table
****
X14$VWR# DS    0CL12          valid work request number
          AGO  .SKIP14
          DC   CL12'WORK#0000001'
          DC   CL12'WORK#0000002'
          DC   CL12'WORK#0000003'
          DC   CL12'WORK#0000004'
          DC   CL12'WORK#0000005'
          DC   CL12'WORK#0000006'
.SKIP14  ANOP
          DC   CL12'CUSTOM000021'  custom work request
          DC   CL12'CUSTOM000022'  custom work request
          DC   CL12'CUSTOM000023'  custom work request
          DC   CL12'CUSTOM000024'  custom work request
X14#VWR# EQU  (*-X14$VWR#)/12     maximum entries

```

- 5 Add comments to your custom code so you will know what you were trying to accomplish when you must reapply your code to a new release of ChangeMan ZMF.

Assemble Exit Source

Stage customized exit source in your ChangeMan ZMF application. If you do not manage ChangeMan ZMF components with ChangeMan ZMF, assemble the exit source manually.

- 1 Use the sample assembly JCL provided in member `ASSEMBLE` in the `CMNZMF CNTL` library, or use your standard assemble procedure.
- 2 Use these options:
 - Assemble: `RENT,ALIGN`
 - Link edit: `RENT, AC=0` (Except `AC=01` for `CMNEX019`)
- 3 Preserve the delivered exit load modules by link editing into a custom load library, not the delivered load library. See ["Preserving Vendor Versions of ChangeMan ZMF Components" on page 16](#).

- 4 Verify that the assemble and link edit steps completed successfully and that the link edit completed without unresolved external references.
- 5 Verify that the link edit options displayed in the directory of your custom LOAD library are the same as those displayed in the directory of the delivered LOAD library.



IMPORTANT! In particular, check directory entries for AC, AM, and RM.

- 6 Test the new exit load module in a test ChangeMan ZMF instance.

Refresh Exit Load

After you assemble and link edit a customized exit, you must take further action to ensure that your changes take effect.

This table tells you what action to take depending on where the exit runs.

| Where the Exit Runs | Action |
|--|--|
| SERNET started task | Stop and start (recycle) the started task |
| ChangeMan ZMF ISPF client | Exit and reenter your ChangeMan ZMF ISPF session |
| File tailoring task (CMNADSP) or batch job | Rerun the task, job, or job step |

This table tells you where each exit runs.

| Exit Name | STC | ISPF | TASK /JOB |
|-----------|-----|------|-----------|
| CMNEXINS | Y | Y | Y |
| CMNEX001 | | Y | |
| CMNEX002 | Y | Y | |
| CMNEX003 | Y | Y | |
| CMNEX004 | Y | | |
| CMNEX005 | | Y | |
| CMNEX006 | Y | Y | |
| CMNEX007 | Y | Y | |
| CMNEX008 | Y | Y | Y |
| CMNEX009 | Y | | Y |
| CMNEX010 | Y | | |
| CMNEX011 | | Y | |
| CMNEX012 | Y | Y | |
| CMNEX014 | Y | Y | |
| CMNEX015 | Y | | |
| CMNEX016 | | | Y |
| CMNEX019 | Y | Y | |
| CMNEX020 | | | Y |

| Exit Name | STC | ISPF | TASK /JOB |
|-----------|-----|------|-----------|
| CMNEX027 | Y | Y | |
| CMNEX028 | Y | Y | Y |
| CMNEX030 | Y | Y | |
| CMNEX031 | Y | | |
| CMNEX032 | Y | | |
| CMNEX033 | | Y | Y |
| CMNEX034 | Y | Y | |
| CMNEX035 | Y | Y | |
| CMNEX036 | | Y | |
| CMNEX037 | | Y | |
| CMNEX038 | | | Y |
| CMNEX039 | | Y | |
| CMNEX040 | | | Y |
| CMNEX041 | | Y | |
| CMNEX042 | | Y | |
| CMNEX043 | Y | Y | |
| CMNEX044 | | | Y |
| CMNEX101 | | | Y |

| Exit Name | STC | ISPF | TASK /JOB |
|-----------|-----|------|-----------|
| CMNEX021 | | | Y |
| CMNEX022 | | | Y |
| CMNEX023 | Y | Y | |
| CMNEX024 | Y | | |
| CMNEX025 | Y | | |
| CMNEX026 | Y | Y | |

| Exit Name | STC | ISPF | TASK /JOB |
|-----------|-----|------|-----------|
| CMNEX102 | Y | | |
| CMNEX103 | | | Y |
| CMNEX201 | | | Y |
| CMNEX210 | Y | | |
| CMNEX220 | | Y | |
| | | | |

Refresh VLF and LLA

If you put ChangeMan ZMF load libraries in the LINKLIST, you must:

- 1 Reload the Virtual Lookaside Facility (VLF) if it is enabled.
- 2 Refresh the Library Lookaside (LLA) facility.



NOTE The *ChangeMan ZMF Installation Guide* recommends that you do not LINKLIST ChangeMan ZMF libraries and that you use STEPLIB and JOBLIB instead.

Exits Listed in SYSPRINT

Active exits are listed in SYSPRINT for the SERNET started task at startup. Example:

```
SER4340I CMNSTART CMNEXINS loaded
SER4340I CMNSTART CMNEX023 loaded
SER4340I CMNSTART CMNEX026 loaded
```



NOTE These four exits are not listed in SYSPRINT even if they are active:

```
CMNEX044
CMNEX201
CMNEX210
CMNEX220
```

Calling XML Services from User Exits

ChangeMan ZMF exit programs can access XML Services by calling client program SERXMLAC. This program is described in an appendix in the *ChangeMan ZMF XML Services User's Guide*.

Customers should be aware that mistakes in their use of SERXMLAC called from an exit can be fatal to the program that called the exit, including SERVER running in the started task.

Exit Descriptions

This section describes ChangeMan ZMF exit functions.

SEREX001

| | |
|-------------------------|---|
| Exit Function | Override the default algorithm that calculates space requirements for reallocating a library that has run out of space in a SERCOPY operation where a compress with IEBCOPY did not remedy the problem. This exit only affects PDS(E) libraries, not PAN, or LIB data sets. |
| Calling Function | Reallocate staging PDS(E) data set (SERNET function) |
| Examples of Use | Use UNIT=SYSDA parameter for reallocated PDS libraries. |

SEREX002

| | |
|-------------------------|---|
| Exit Function | Validate or alter the first four JOB statement records in JCL before the job is submitted for execution. |
| Calling Function | Module SERSUBMT calls this exit each time a job is submitted |
| Notes | <p>This exit runs in the SERNET started task address space, which provides security against a malicious user being able to submit jobs. See exit CMNEX008, which performs a similar function but executes in the address space of the user who submits a job rather than in the SERNET started task address space.</p> <p>This exit, as delivered, is enabled but does not perform any JOB statement alterations. Follow the comments in the source code if you want to alter JOB statements or disable the exit.</p> |

SEREX003

| | |
|----------------------|---|
| Exit Function | Allows read-only access to JES output, and only allows users to cancel, purge, or requeue jobs that they own. As delivered, CMNEX003 is enabled. |
| Notes | <p>IMPORTANT! Access to JES jobs is normally controlled by resource classes JESJOBS and JESSPOOL, regardless of whether SEREX003 is activated. If these resource classes are activated and appropriate rules have been established by your security administrator, we recommend that you disable this exit. To disable the exit, do one of the following:</p> <ul style="list-style-type: none"> ■ Use SERNET keyword parameter EX003=NO. ■ Customize the exit as described in source code comments at the top of the program. |

SEREX005

| | |
|-------------------------|---|
| Exit Function | Provide library member level security. After a library name is checked against data set access rules in your security system, SEREX005 constructs a second data set name that is checked in your security system: Original DSN LIBRARY.NAME SEREX005 DSN LIBRARY.NAME.MEMBER |
| Calling Function | Various ChangeMan ZMF modules call this exit for every GET, PUT, DElete, or RENAME library member request. |
| Examples of Use | To allow userIDs in group \$ABC exclusive update access to LIBRARY.NAME members that have names starting with XYZ, leave SEREX005 enabled and issue these RACF commands: ADDSD 'LIBRARY.NAME*.*' UACC(READ) PERMIT 'LIBRARY.NAME*.XYZ*' ID(\$ABC) ACCESS(UPDATE) |
| Notes | For this exit to function correctly, the length of the constructed data set name LIBRARY.NAME.MEMBER cannot exceed 44 characters. To avoid this limitation and to restrict access to components within the framework of ZMF functions and security, we recommend that you disable CMNEX005 and use ZMF component level security. See topic "Setting Component Level Security" in the <i>ChangeMan ZMF Administrator's Guide</i> . IMPORTANT! As delivered, CMNEX005 is enabled. To disable SEREX005, do one of the following: <ul style="list-style-type: none"> ■ Use SERNET keyword parameter EX005=NO. ■ Customize the exit as described in source code comments at the top of the program. |

CMNEXINS

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|-------------------------|---|------------------------------|------------------------|---|---------------------------------|-----------------------------------|-------------------------|-------------------------|-------------------------------|---|-----------------|--------------------------------|----------------------|---------------------------|-----------|------------------------------------|--------|----------------------------------|------------------|---|-----------------|-------------------------|-------------------------|--|--------------------------------------|--|--------------------------------|
| Exit Function | Specify the data set name, space allocation, DCB attributes, and extended attributes for EAV (extended address volumes) of the temporary work data set used during logon, checkout, edit, stage, recompile, relink, file tailoring, browse, compare, monitor, and many utility functions. | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Calling Function | <table border="0"> <tr> <td>API Service - Component</td> <td>Copy VSAM Package Master to sequential file</td> </tr> <tr> <td>API Service - File tailoring</td> <td>Edit package component</td> </tr> <tr> <td>API Service - Installation job file tailoring</td> <td>Edit / Browse Notification File</td> </tr> <tr> <td>Batch I/A table maintenance Batch</td> <td>ISPF client main driver</td> </tr> <tr> <td>Interface - Main driver</td> <td>Monitor (Limbo and Scheduler)</td> </tr> <tr> <td>Batch Interface - Main driver (freeze, stage)</td> <td>Promote package</td> </tr> <tr> <td>Batch stage called from CMNAPI</td> <td>Query Package Master</td> </tr> <tr> <td>Browse compressed listing</td> <td>Recompile</td> </tr> <tr> <td>Browse/print/copy baseline members</td> <td>Relink</td> </tr> <tr> <td>Checkout from baseline/promotion</td> <td>Staging Versions</td> </tr> <tr> <td>Compare staging library to baseline/promotion</td> <td>Submit services</td> </tr> <tr> <td>Copy / Stage components</td> <td>XML Services - Data set</td> </tr> <tr> <td></td> <td>XML Services - Save Staging Versions</td> </tr> <tr> <td></td> <td>XML Services - Utility Request</td> </tr> </table> | API Service - Component | Copy VSAM Package Master to sequential file | API Service - File tailoring | Edit package component | API Service - Installation job file tailoring | Edit / Browse Notification File | Batch I/A table maintenance Batch | ISPF client main driver | Interface - Main driver | Monitor (Limbo and Scheduler) | Batch Interface - Main driver (freeze, stage) | Promote package | Batch stage called from CMNAPI | Query Package Master | Browse compressed listing | Recompile | Browse/print/copy baseline members | Relink | Checkout from baseline/promotion | Staging Versions | Compare staging library to baseline/promotion | Submit services | Copy / Stage components | XML Services - Data set | | XML Services - Save Staging Versions | | XML Services - Utility Request |
| API Service - Component | Copy VSAM Package Master to sequential file | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| API Service - File tailoring | Edit package component | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| API Service - Installation job file tailoring | Edit / Browse Notification File | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Batch I/A table maintenance Batch | ISPF client main driver | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Interface - Main driver | Monitor (Limbo and Scheduler) | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Batch Interface - Main driver (freeze, stage) | Promote package | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Batch stage called from CMNAPI | Query Package Master | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Browse compressed listing | Recompile | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Browse/print/copy baseline members | Relink | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Checkout from baseline/promotion | Staging Versions | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Compare staging library to baseline/promotion | Submit services | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Copy / Stage components | XML Services - Data set | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | XML Services - Save Staging Versions | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | XML Services - Utility Request | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | |
|------------------------|--|
| Examples of Use | Most customers use this exit to set a high level qualifier so that ChangeMan ZMF does not need ALTER access to every TSO user's high level qualifier. |
| Notes | <p>Sample code between AGO <i>.label</i> and <i>.label</i> ANOP statements is skipped by the assembler and does not appear in the assembler listing. To enable the code, delete or comment out the AGO statement.</p> <p>A particular volume serial for temporary work data sets can be specified in several locations in ChangeMan ZMF. This is the search order for a specified volume serial:</p> <ol style="list-style-type: none"> 1 X15\$VOLS in CMNEX015 if this exit is enabled 2 INS\$VOLS in CMNEXINS if this exit is enabled 3 DEFAULT VOLUME SERIAL field in Application Administration Parameters 4 DEFAULT VOLUME SERIAL field in Global Administration Parameters |

CMNEX001

| | |
|-------------------------|---|
| Exit Function | Specify who can update package information. |
| Calling Function | Update change package information |
| Examples of Use | <p>Allow anyone to update package information before first approval entered.</p> <p>Allow approvers who have not approved frozen package to update installation date.</p> <p>Block changes to super or complex packages except by administrators.</p> |

CMNEX002

| | |
|-------------------------|--|
| Exit Function | Restrict installation date by one of more of the following: Application, global or application administrator authority, specified date, day of week, package type, from install time, to install time, values in package user information variables. |
| Calling Function | <p>Create change package</p> <p>Update package installation date</p> |
| Examples of Use | Restrict CICS changes to weekend installation dates. |

CMNEX003

| | |
|-------------------------|---|
| Exit Function | Impose a lead time for package create from the installation date using one or more of the following: Application, global or application administrator authority, specified date, day of week, package type, values in package user information variables. |
| Calling Function | <p>Create change package</p> <p>Update package information</p> |
| Examples of Use | <p>Enforce installation lead time standards for packages with high risk rating.</p> <p>Block planned package installation date lead time less than 3 business days.</p> |

CMNEX004

| | |
|-------------------------|--|
| Exit Function | Prohibit use of specified mnemonics when creating applications. |
| Calling Function | Application Administration: Create new application ChangeMan ZMF |

CMNEX005

| | |
|-------------------------|--|
| Exit Function | Set requirements for application approval lists. |
| Calling Function | Application Administration: Create/update planned and unplanned approval lists |
| Examples of Use | Require at least two approvals for an unplanned approval list. Validate approval security entity naming conventions against application mnemonic. |

CMNEX006

| | |
|-------------------------|--|
| Exit Function | Restrict creation of packages with specified package level by application. |
| Calling Function | Create change package |
| Examples of Use | Prohibit super/complex packages in specified applications. |

CMNEX007

| | |
|-------------------------|---|
| Exit Function | Restrict package installation date by administrator authority, application, user, installation date, package type, or package user information variables. |
| Calling Function | Create change package Update change package information |

CMNEX008

| | |
|-------------------------|---|
| Exit Function | Validate or modify JOB statement information in the first four batch job records. |
| Calling Function | Submit batch job from ChangeMan ZMF online function Build package installation JCL (X Node data set) Build remote promote JCL |
| Examples of Use | Prohibit use of JCLLIB, EXEC, and INCLUDE keywords in JCL coded in JOB statement Information fields on user and administration panels. Override or restructure job names obtained from ChangeMan ZMF administration. |

CMNEX009

| | |
|-------------------------|---|
| Exit Function | Add approvals to the Planned Approval List during the package freeze or post-approval processes. Approvals can be added based on one or more of the following conditions: <ul style="list-style-type: none"> ■ If specified library types are included in the package and have components staged into them ■ If installations are scheduled at specified remote sites in the package ■ If specified library types have scratched and/or renamed components included in the package |
| Calling Function | Freeze package |
| Examples of Use | Add CIO approval if package user information indicates a high risk change. Add CICS systems programmer as approver if package contains BMS Map change. Add data center manager approver if package will be installed at remote center. Add a DBA approver if a package contains scratched DBRM components. |
| Notes | Copybook CMNEX009 contains library type table X09\$SLTP, with up to 100 entries, that tells you what library types are in the package. Each entry in the table contains indicator X09\$FTYP with three independent bit switches to tell you if any one of the package components in the library type is a staged component, a scratch utility request, or a rename utility request. The global administrator must set the Display Package User Option Panel(s) field to YES on the Global Parameters - Part 4 of 6 (CMNGGP04) panel if you want to be able to access user variables <i>user0101</i> through <i>user7205</i> from this exit. |

The points in the package lifecycle at which CMNEX009 is called depends on whether a package is planned or unplanned.

For planned packages, CMNEX009 is called at package freeze if you have requested it to check for any of the criteria listed above. Approvers that you designate in the exit code are notified in addition to the approvers on the planned package list. Once approved, the package is installed at the scheduled time.

For unplanned packages, CMNEX009 is called at two points in the package lifecycle:

- At package freeze - Approvers that you designate in the exit code are notified in addition to the approvers on the unplanned package list.
- At post-approval processing (after the package has been installed) - The approver list is rebuilt. Approvers that you designate in the exit code are added to the planned package approver list and are notified in addition to the approvers on the planned list after the package has been installed.

Consult the comments in the source code for this exit for detailed instructions on how to code the conditions that the exit is to check and to add approvers if these conditions are true.

CMNEX010

| | |
|-------------------------|---|
| Exit Function | Restrict who can enter a package approval by one or more of the following: Global or administration authority, package name, approving userid, date of approval, subsystem ID, package type, package level, approval function (A,R,C.V), approval entity, work request number, package creator, hierarchical order number, nearest installation date. |
| Calling Function | Approve package |
| Examples of Use | Prevent a package creator from approving any package, planned or unplanned. |
| Notes | This exit is called after an approval panel has been submitted by the user but before the approval is processed. |

CMNEX011

| | |
|-------------------------|--|
| Exit Function | Set minimum library security access requirements for promotion, baseline, and production library configuration functions. |
| Calling Function | Configure application libraries in application administration |
| Examples of Use | Prevent configuration of a library for promotion if the ChangeMan ZMF instance does not have update access to the library. |

CMNEX012

| | |
|-------------------------|--|
| Exit Function | Compare the package create day-of-week to the specified normal business days-of-week to determine whether to use the unplanned approval list for unplanned packages. |
| Calling Function | Create package |
| Examples of Use | Consider Wednesday to be a non-business day so that unplanned packages created on Wednesday are always defined with the unplanned approval list. |

CMNEX014

| | |
|-------------------------|---|
| Exit Function | Validate department number and/or work request number against a specified list of valid values by application. Cross-edit work request number, department number, and package user information when flag X14\$IVAL indicates that package user information is passed. |
| Calling Function | Create package Update package information Create package user information Update package user information |

CMNEX015

| | |
|-------------------------|--|
| Exit Function | Direct dynamically allocated temporary work data sets to a specified VOLSER. |
| Calling Function | Any task that involves the dynamic allocation of data sets |
| Notes | <p>A specific volume serial for temporary work data sets can be specified in several locations in ChangeMan ZMF. This is the search order for a specified volume serial:</p> <ol style="list-style-type: none"> 1 X15\$VOLS in CMNEX015 if this exit is enabled 2 INS\$VOLS in CMNEXINS if this exit is enabled 3 DEFAULT VOLUME SERIAL field in Application Administration Parameters 4 DEFAULT VOLUME SERIAL field in Global Administration Parameters |

CMNEX016

| | |
|-------------------------|---|
| Exit Function | Modify or skip expanded copybook records written to SYSOFILE by CMNWRITE. |
| Calling Function | Program CMNWRITE in build jobs |
| Examples of Use | Create a PANVALET like exit to prefix COBOL COPYBOOK source statements with user defined input. |
| Notes | <p>CMNEX016 mimics a CA Librarian exit that modifies copybook records included in source with the -INC command.</p> <p>Sample code between AGO <i>.label</i> and <i>.label</i> ANOP statements is skipped by the assembler and does not appear in the assembler listing. To enable the code, delete or comment out the AGO statement.</p> |

CMNEX019

| | |
|-------------------------|--|
| Exit Function | This exit is called before and after checkout, checkin, and build and may be used for any function desired at those points in component processing, depending on the availability of data passed from and returned to the calling program. |
| Calling Function | Checkout component Checkin component Build component |
| Notes | <p>Checkin is one of the two processes that make up the stage from development function:</p> <ul style="list-style-type: none"> ■ Checkin copies a component from a library outside ZMF to a staging library. ■ Build transforms source into one or more executables. <p>Sample code between AGO <i>.label</i> and <i>.label</i> ANOP statements is skipped by the assembler and does not appear in the assembler listing. To enable the code, delete or comment out the AGO statement.</p> <p>This exit was used with the ChangeMan ZMF APS Option that was retired with Version 6.1.</p> |

CMNEX020

| | |
|-------------------------|--|
| Exit Function | Reset return codes for package audit out-of-synch conditions. |
| Calling Function | Package Audit |
| Notes | Sample code between AGO <i>.label</i> and <i>.label</i> ANOP statements is skipped by the assembler and does not appear in the assembler listing. To enable the code, delete or comment out the AGO statement. |

CMNEX021

| | |
|-------------------------|--|
| Exit Function | Use library type, application, package creator, or other fields passed from package audit to bypass: <ul style="list-style-type: none"> ■ SYNCHnn! processing ■ Component relationship processing ■ Promotion libraries in SYNCH15! processing. |
| Calling Function | Package Audit |
| Notes | Sample code between AGO <i>.label</i> and <i>.label</i> ANOP statements is skipped by the assembler and does not appear in the assembler listing. To enable the code, delete or comment out the AGO statement. |

CMNEX022

| | |
|-------------------------|---|
| Exit Function | Exclude specified load and non-load components from processing by package audit and by the impact analysis LDS build. Include CSECT, with the same name as the composite load module, in the Impact Analysis Table to show LOD relationships, and in package audit processing to detect SYNCH8!. |
| Calling Function | Package Audit Impact Analysis Maintenance |
| Examples of Use | Prevent IBM subroutines with names starting in ILBO, DLI, DFS, DFH, CEE, IBM, and IGZ from cluttering up the package audit report where they obscure important information. |

CMNEX023

| | |
|-------------------------|--|
| Exit Function | Enable the Package User Information facility. Define ISPF variable names used for Package User Information variables in file tailoring for installation JCL. |
| Calling Function | All remote file tailoring and ISPF driven stage, recompile, and relink package functions |

CMNEX024

| | |
|-------------------------|---|
| Exit Function | Prohibit package freeze depending on one of more of the following: Application, global or application authority, package type, library types in package, installation date, installation from or to time, installation day of week, Package User Information variable. |
| Calling Function | Freeze change package |

CMNEX025

| | |
|-------------------------|---|
| Exit Function | Prohibit package freeze or selective refreeze of component depending on compile parameters, link edit parameters, and user options used in last stage job. Other validations can be performed on application mnemonic, component library types, component names, language, compile procedure. |
| Calling Function | Freeze change package Selective refreeze |

CMNEX026

| | |
|-------------------------|---|
| Exit Function | Dynamically allocate additional staging libraries based on component type, user options, or other data passed to the exit. Prohibit package component delete based on library type, member name, or whether component is promoted. |
| Calling Function | Stage component Delete component during staging |
| Examples of Use | Allocate additional staging libraries for build processes with multiple outputs. |
| Notes | Do not use CMNEX026 to control delete of related components from staging libraries. Use ILOD records instead. |

CMNEX027

| | |
|-------------------------|---|
| Exit Function | Override the promotion or demotion rule for individual promotion levels. |
| Calling Function | Promote package Demote package Unfreeze/refreeze Revert |
| Examples of Use | Specify Promotion Rule by Promotion Site/Level Break down Promotion Rules into behaviors that can be specified separately for a site/Level. |
| Notes | CMNEX027 is called twice for promotion/demotion: <ul style="list-style-type: none"> ■ Before the promotion or demotion panel displays the history ■ After promotion or demotion |

CMNEX028

| | |
|-------------------------|---|
| Exit Function | Set several administration options that are not included in the ISPF interface: <ul style="list-style-type: none"> ■ Add extra job statements for batch processing ■ Do not to release unused space in staging libraries during a freeze ■ Bypass CMNAPI case conversion for lower case languages ■ Bypass expansion of duplicate %INCLUDE statements when the copybook does not exist in the staging library |
| Calling Function | Extended administration options affecting various areas throughout ChangeMan ZMF |
| Notes | JOB statements 5 and 6 added for batch recompile, batch freeze, and package audit only. |

CMNEX030

| | |
|-------------------------|---|
| Exit Function | Bypass checkout enforcement rule depending on the component library type. |
| Calling Function | Stage component from development |

CMNEX031

| | |
|-------------------------|--|
| Exit Function | Bypass package audit processing based on library type. |
| Calling Function | Package audit Package integrity check |

CMNEX032

| | |
|-------------------------|---|
| Exit Function | Process specified library types as compressed listings (like-LST) |
| Calling Function | Baseline browse |

CMNEX033

| | |
|-------------------------|---|
| Exit Function | Override package status validation when adding or removing participating packages in complex or super packages. Allow automatic close of super or complex packages after update if all participating packages have completed the package lifecycle. |
| Calling Function | Update Complex/Super Information Close super or complex packages |

CMNEX034

| | |
|-------------------------|--|
| Exit Function | Assign processing to specified library types when creating or updating impact analysis relationships. Processing that can be assigned with CMNEX034: JCL Process the specified library type as JCL when creating JCL-Procedure, PGM Name/Symbol, and DSN Name/Symbol relationship records. PRC Process the specified library type as cataloged procedures when creating JCL-Procedure, Pgm Name/Symbol, and DSN Name/Symbol relationship records. |
| Calling Function | Impact Analysis Maintenance Baseline ripple |
| Examples of Use | Parse components in library type JC2 like JCL to create relationship records in the Impact Analysis file. |
| Notes | Starting with ChangeMan ZMF 6.1, object components and NCAL load components are supported in the base product with like-object (O) and like-NCAL (N) in global and application administration. |

CMNEX035

| | |
|-------------------------|--|
| Exit Function | Restrict the library types displayed on the valid library selection list for checkout, stage, browse baseline, browse compressed listing, compare, scan, and scratch/rename functions. |
| Calling Function | Checkout, stage, browse baseline, browse compressed listing, compare, scan, scratch/rename |

CMNEX036

| | |
|-------------------------|--|
| Exit Function | Call an edit preprocessor or different editor. |
| Calling Function | Edit-in-stage |
| Examples of Use | Use SMART EDIT instead of ISREDIT. Invoke an SDF2 editor interface. |

CMNEX037

| | |
|-------------------------|--|
| Exit Function | Call an edit macro like ASG-JCLPREP or your own edit macro. |
| Calling Function | Edit-in-stage VIEW in edit-in-stage |
| Examples of Use | Call edit macro for ASG-JCLPREP at end of edit-in-stage session for JCL members. |

CMNEX038

| | |
|-------------------------|---|
| Exit Function | Assign a default language for a particular library type. Specify a starting column and ending column for source code parsing by CMNPARSE. |
| Calling Function | Parse source to determine language |
| Notes | The starting column must start before the ending column. The copy/include statement must fit between the starting and ending columns. |

CMNEX039

| | |
|-------------------------|--|
| Exit Function | Assign group names to a list of installation sites. Installation site group names are displayed on the Create: Site Information panel and Update: Site Information panel, and on the Site Selection List panel for those functions. The Query: Site Information panel shows individual site names, including any sites in an installation site group. |
| Calling Function | Create package Update package information |
| Notes | If an installation site group includes a site that is not fully defined in an application, the group name is not displayed in package create or update in that application. Caution! If an installation site is included in more than one installation site group in CMNEX039, and both groups are added to a package, then the site is added to the package twice, and file tailoring for package installation JCL will fail with messages CMN1000A and CMN8703I. |

CMNEX040

| | |
|-------------------------|---|
| Exit Function | Enforce the use of a specific file or PDS member for package audit auto resolve parameters. |
| Calling Function | Package audit |

CMNEX041

| | |
|-------------------------|---|
| Exit Function | Specify special authorization for package IMS information update. |
| Calling Function | Updating IMS package information |

CMNEX042

| | |
|-------------------------|--|
| Exit Function | Display panel CMNEX042 for component general description and store information as discrete fields on the package master. |
| Calling Function | Stage component |

CMNEX043

| | |
|-------------------------|---|
| Exit Function | Add custom processes that are executed outside of ChangeMan ZMF at the end of package create. |
| Calling Function | Create change package |

CMNEX044

| | |
|-------------------------|---|
| Exit Function | Specify like-copy library types to be excluded from package audit SYNCH15! processing. |
| Calling Function | Package audit |
| Notes | See exit CMNEX021 to bypass promotion libraries in the processing of SYNCH15! conditions. |

CMNEX093

| | |
|-------------------------|--|
| Exit Function | Override the default 755 permission for HFS staging libraries. |
| Calling Function | This exit is loaded at startup and is called by LIBTYPE SERVICE ALLOCATE during the dynamic allocation of an HFS staging library (HFS path). |

CMNEX101

| | |
|-------------------------|---|
| Exit Function | Add, manipulate, or verify DB2 bind control processed by CMNDB2PL. |
| Calling Function | DB2 Option binds at promotion, installation, and baseline ripple. This exit is called from the DB2 Option Plan Lookup Program CMNDB2PL. |
| Notes | CMNEX101 is called after DSN BIND commands are templated by CMNDB2PL and non-standard records set aside by CMNEX103 are restored to the end of the command set. BIND command sets are presented to CMNEX101 one command keyword per record. |

CMNEX102

| | |
|-------------------------|--|
| Exit Function | Define a collection ID for the DB2 Option that is different from the default of CMNx (where x is the Subsystem ID of the ChangeMan ZMF instance). |
| Calling Function | Program CMNDB2SQ in all DB2 Option functions that access a ChangeMan ZMF collection ID under CMNPLAN, including plan lookups for DB2 binds at promotion, installation, and baseline ripple |

CMNEX103

| | |
|-------------------------|---|
| Exit Function | Sets aside records from DSN BIND command sets before parsing in CMNDB2PL in preparation for templating. |
| Calling Function | Program CMNDB2PL in the DB2 Option before keyword operand templating. |

| | |
|------------------------|---|
| Examples of Use | Set aside "comment" records with asterisk (*) in position 1 that control the behavior of CMNEX101, then restore the comment records at the bottom of the DSN BIND command or stored procedure DDL after templating. |
| Notes | Exit program CMNEX103 is called by program CMNDB2PL to delete or set aside records in DSN BIND commands. Using CMNEX103, you can take one of three actions for each DSN BIND: <ul style="list-style-type: none"> ■ Pass the record back for parsing and keyword operand templating. ■ Drop the record. ■ Withhold the record from parsing and templating, then restore the record at the end of the DSN BIND command set before CMNEX101 is called. |

CMNEX201

| | |
|-------------------------|--|
| Exit Function | Bypass processing of specified library types in ERO release audit. |
| Calling Function | ERO Release Audit |
| Examples of Use | Exclude a DDI library type from release audit processing. |

CMNEX210

| | |
|-------------------------|---|
| Exit Function | Validate conditions before attaching a package to an ERO release or detaching a package from a release. |
| Calling Function | Attach a package to a release Detach a package from a release |
| Notes | If activated, this exit is called before the attach or detach function and after the attach or detach function. On the pre-function call anything but an RC=0 will halt the function. If the package is already attached to a release and it is being updated to a new release, the exit is actually called four times: a pre- and post-function call for detaching the package from the old release and a pre- and post-function call for attaching the package to the new release. Only users with global, application, or release administrator authority can detach packages from a release. |

CMNEX220

| | |
|-------------------------|--|
| Exit Function | Validate conditions before checking in a component to an ERO release area or before retrieving a component from an area. |
| Calling Function | Check components in to an ERO release Retrieve components from a release |
| Notes | Pre-checkin or pre-retrieve calls can alter the selected component list or halt all the processing. The components that are selected by this exit are skipped by the check-in or retrieve function. All other components are selected as normal. Post-checkin or post-retrieve calls to this exit have no effect on the internal processing of the check-in or retrieve operation. On post-check-in or post-retrieve calls, the component list only contains components that completed the function successfully. |

Chapter 4

User Data

ChangeMan ZMF includes four facilities that enable you to enter information that can be used by skeleton file tailoring to customize ChangeMan ZMF functions that are executed in batch jobs. Some of these facilities store your data in the package master or component master files, and you can display that information on custom reports.

| | |
|--|----|
| Package User Information | 60 |
| Staging User Options | 64 |
| Release ID Variables | 71 |
| Custom V01-V10 Variables | 77 |
| Summary | 79 |

Package User Information

Package User Information is an optional facility that stores data in 71 fields of various lengths on the package master. You enter Package User Information on panels that are displayed when you create a package and when you update package information. The information stored in Package User Information fields is available for processing by several exits, and it is available in file tailoring for installation JCL.

The Package User Information facility is designed to be flexible so that you can customize it to meet your needs for package level user data. You can customize up to two data entry panels, selecting the fields you want to display, labeling the input fields with names you choose, and coding edit rules and other panel processing to satisfy your requirements. You can use Package User Information data in program logic in certain ChangeMan ZMF exit programs. You can choose your own names for the variables that are made available to file tailoring for install JCL.

Package user information is available in all remote file tailoring and ISPF driven stage, recompile and relink package functions, and all exit calls which are package driven.

Package User Information Field Names

Package User Information fields have different names on input panels, in copybook CMNEXITS that represents how they are stored on the package master, and in file tailoring for install JCL. The field names follow a convention that relates the name representing the data stored on the package master to the other names that refer to the same data. This naming convention also tells you how long the data field is.

The following example shows the names that identify a 3-byte Package User Information field in various ChangeMan ZMF functions. In the field naming convention:

- *ll* represents the length of the field in bytes.
- *nn* is a field identifier that is unique among fields of the same length.

| Where Name is Used | Name | Modifiable? | Example |
|--------------------|-------------------|-------------|-----------|
| Input Panels | USR <i>llnn</i> | No | USR0301 |
| Copybook CMNEXITS | IXP\$ <i>llnn</i> | No | IXP\$0301 |
| File Tailoring | USR <i>llnn</i> | Yes | USR0301 |

There are a total of 71 Package User Information fields. They vary in length from 1 byte to 72 bytes. This table shows how many fields of each length are stored on the package master:

| Field Length | Count | Field Names on ISPF Panels | CMNEXITS Field Names (Package Master) | Default File Tailoring Variable Names | ZDDOPTS Variable Names |
|--------------|-------|----------------------------|---------------------------------------|---------------------------------------|-------------------------------|
| 1 | 15 | USR0101 to USR0115 | IXP\$0101 to IXP\$0115 | USR0101 to USR0115 | UserVarLen101 - UserVarLen115 |
| 2 | 11 | USR0201 to USR0211 | IXP\$0201 to IXP\$0211 | USR0201 to USR0211 | UserVarLen201 - UserVarLen211 |
| 3 | 10 | USR0301 to USR0310 | IXP\$0301 to IXP\$0310 | USR0301 to USR0310 | UserVarLen301 - UserVarLen310 |

| Field Length | Count | Field Names on ISPF Panels | CMNEXITS Field Names (Package Master) | Default File Tailoring Variable Names | ZDDOPTS Variable Names |
|--------------|-------|----------------------------|---------------------------------------|---------------------------------------|---------------------------------|
| 4 | 10 | USR0401 to USR0410 | IXP\$0401 to IXP\$0410 | USR0401 to USR0410 | UserVarLen401 - UserVarLen410 |
| 8 | 10 | USR0801 to USR0810 | IXP\$0801 to IXP\$0810 | USR0801 to USR0810 | UserVarLen801 - UserVarLen810 |
| 16 | 5 | USR1601 to USR1605 | IXP\$1601 to IXP\$1605 | USR1601 to USR1605 | UserVarLen1601 - UserVarLen1605 |
| 44 | 5 | USR4401 to USR4405 | IXP\$4401 to IXP\$4405 | USR4401 to USR4405 | UserVarLen4401 - UserVarLen4405 |
| 72 | 5 | USR7201 to USR7205 | IXP\$7201 to IXP\$7205 | USR7201 to USR7205 | UserVarLen7201 - UserVarLen7205 |

The last column of the table show the default names for variables made available to file tailoring for install JCL. You can change the default variable names to names that are meaningful to you.

Package User Information Input Panels

Your ChangeMan ZMF global administrator activates the Package User Information feature by selecting the following option on the Global Parameters - Part 5 of 8 (CMNGGP05) panel:

_ Enable package user variables

If the Package User Information feature is activated, two sample ISPF input panels are displayed when you create a change package or update package information.

You can customize these panels to display and process the Package User Information fields that you want to use to store information on the package master.

The first Package User Information panel displayed is the **Create - Sample Package User Panel 1** (CMNDPUP1).

```

CMNDPUP1          CREATE - Sample Package User Panel 1
Command ===> _____

Enter "yes" or "no" to indicate value of variable

Field 1 . . . . . NO
Field 2 . . . . . NO
Field 3 . . . . . NO
Field 4 . . . . . NO
Field 5 . . . . . NO
Field 6 . . . . . NO

Enter "/" to select option
  / Next panel

```

If you select the **Next panel** option on the **Create - Sample Package User Panel 1**, the **Create - Sample Package User Panel 2** (CMNDPUP2) is displayed.

```

CMNDPUP2          CREATE - Sample Package User Panel 2
Command ===> _____

Enter "yes" or "no" to indicate value of variable:

Test Value1 . . . . . NO
Test Value2 . . . . . NO
Test Value3 . . . . . NO
Test Value4 . . . . . NO

```

Package User Information and Exits

Package User Information fields are included in copybook CMNEXITS and are available in every package related exit program.

These exits are for ChangeMan ZMF basic package functions:

| | | | | |
|----------|----------|----------|----------|----------|
| CMNEX001 | CMNEX009 | CMNEX024 | CMNEX029 | CMNEX038 |
| CMNEX002 | CMNEX010 | CMNEX025 | CMNEX030 | CMNEX041 |
| CMNEX003 | CMNEX014 | CMNEX026 | CMNEX033 | CMNEX043 |
| CMNEX007 | CMNEX019 | CMNEX027 | CMNEX036 | |
| CMNEX008 | CMNEX023 | CMNEX028 | CMNEX037 | |

For package audit, package user information is available in exits CMNEX020, CMNEX021, and CMNEX031.

For ERO, package user information is available in exit CMNEX210.

Implementing the Package User Information Facility

Follow the steps in this section to modify Package User Information components delivered in ChangeMan ZMF libraries so that they satisfy your needs for package level user data.

Choose Package User Information Fields

- 1 List the kind of data you want to store for a change package and map your list to the 71 fields available in the Package User Information facility.
- 2 Choose the shortest fields that will accommodate the data that you will store.

See ["Package User Information Field Names" on page 60](#) for a description of the 71 available fields.

Modify Sample Package User Information Panels

- 1 Copy sample Package User Information panels CMNDPUP1 and CMNDPUP2 to your custom panels library from the CMNZMF PANELS library unloaded from the ZMF installer.

- 2 Modify the **)BODY** section of the panels to change the title displayed on the panels, and to display field tags that identify the data users should enter.
- 3 Modify the **)INIT** section to initialize blank fields. Modify the ZVARS statement to associate panel fields with the appropriate Package User Information field name.
- 4 Modify the **)PROC** section to validate information entered by the user and to perform cross-field edits, if required.



NOTE Panel field name USR0199 is reserved for the **Next Panel** field on panel CMNDPUP1 to determine whether the second Package User Information panel CMNDPUP2 will be displayed. Even if you do not use panel CMNDPUP2, and alter the panel so that you do not display field USR0199 on CMNDPUP1, you must still set a value for this field.

- 5 Copy sample help panels CMN12350 and CMN12355 to your custom panels library from the CMNZMF PANELS library unloaded from the ZMF installer.
- 6 Modify the panels to describe the fields on your custom CMNDPUP1 and CMNDPUP2 panels respectively.

Modify Exits

Package User Information fields in copybook CMNEXITS are populated with your data stored on the package master when the exit is called.

- 1 Copy the exit program source you want to modify to your custom source library from the CMNZMF ASMSRC library delivered in the ChangeMan ZMF installer.
- 2 Modify exit program logic to use the Package User Information fields in copybook CMNEXITS.

See [Chapter 3, "User Exits" on page 39](#) for general instructions for enabling and coding ChangeMan ZMF exit programs. Coding for exit program CMNEX023 is described in the next topic.

Modify Exit 23 For Install JCL File Tailoring

If you want to use Package User Information in file tailoring for install JCL, you must enable exit program CMNEX023. This exit defines ISPF variables for Package User Information in the ISPF session used by file tailoring.

You can use CMNEX023 to change the names of the ISPF variables that are defined in the ISPF session for install JCL file tailoring. You can also use CMNEX023 to populate variables with other information available to the exit.

The sample code delivered in CMNEX023 shows modifications to accomplish both of these objectives. For example, the default file tailoring variable USR0115 is renamed to

X23PTYP, and whatever data was stored on the package master for that field is overlaid with the Package Type that is in CMNEXITS field IXP\$PTYP.



NOTE When you enable exit program CMNEX023 as it is delivered in the ZMF installer, some Package User Information variables defined to file tailoring for install JCL will be modified by the sample code in the exit. Check the program comments, the variable names in #SPFVARS, and the procedure code at label EXT\$0000 to ensure that the sample code will not interfere with what you want to do in file tailoring for install JCL. You may have to change the sample code to restore the default Package User Information ISPF variable names and field contents.

Modify Install Skeletons

- 1 Copy install skeletons to your custom skeleton library from the CMNZMF SKELS library unloaded from the ZMF installer.
- 2 Modify those skeletons to use the ISPF variables you defined in exit program CMNEX023.

Enable Package User Information

In Global Administration Parameters (=A.G.1), select the **Enable package user variables** field on the **Global Parameters - Part 5 of 8** panel (CMNGGP05):

```

CMNGGP05                Global Parameters - Part 5 of 8
Command ===> _____

Audit package lock . . . . OPTIONAL (Always/Never/Optional)

Enter "/" to select option
  _ Job name increment override
  _ Use zprefix in batch jobs
  _ Suppress msgs in dis/ins/bas jobs
  _ Create component work records
  _ Force audit of unplanned packages
  _ Allow link packages
  _ Memo delete empty packages only
  / Enable package user variables
  _ Enable component user variables
  _ Add user variables to package list table
  _ Allow component in multiple applications
  _ Auto scratch load member with source
Approval Restrictions
  _ Package creator cannot approve
  _ Package worker cannot approve
  _ Only 1 approval per user

```

Staging User Options

User options are component-level user data that is stored in 57 fields of various lengths on the component master. You enter user options on customizable panels that are displayed in build processes like stage, recompile and relink. The information stored in user options is available in file tailoring for build processing JCL.

User option settings are included in designated compile procedures, so you can lock down these fields that determine how build processing is performed. User options can be validated by exit program CMNEX025 to ensure that prohibited values are not used before a component is installed.

Using a combination of user options and custom compile procedure skeletons, you can create highly flexible build processes that fits your unique needs.

User Options Field Names

The 57 user option fields vary in length from 1 byte to 72 bytes. This table shows user option field lengths and the names of the fields at key points in component build processing:

| Field Length | Count | Field Names on ISPF Panels | CMNEX025 Field Names | CMNEX026 Field Names | File Tailoring Variable Names | ZDDOPTS Variable Names |
|--------------|-------|----------------------------|------------------------|------------------------|-------------------------------|----------------------------------|
| 1 | 20 | USROP01 to USROP20 | X25\$U001 to X25\$U020 | X26\$OP01 to X26\$OP20 | USROP01 to USROP20 | UserOption01 to UserOption20 |
| 1 | 5 | CUSR011 to CUSR015 | X25\$0101 to X25\$0105 | | CUSR011 to CUSR015 | UserOption101 to UserOption105 |
| 2 | 3 | CUSR021 to CUSR023 | X25\$0201 to X25\$0203 | | CUSR021 to CUSR023 | UserOption201 to UserOption223 |
| 3 | 3 | CUSR031 to CUSR033 | X25\$0301 to X25\$0303 | | CUSR031 to CUSR033 | UserOption301 to UserOption303 |
| 4 | 3 | CUSR041 to CUSR043 | X25\$0401 to X25\$0403 | | CUSR041 to CUSR043 | UserOption401 to UserOption403 |
| 8 | 5 | CUSR081 to CUSR085 | X25\$0801 to X25\$0805 | | CUSR081 to CUSR085 | UserOption801 to UserOption805 |
| 10 | 2 | CUSR101 to CUSR102 | X25\$1001 to X25\$1002 | | CUSR101 to CUSR102 | UserOption1001 to UserOption1002 |
| 16 | 2 | CUSR161 to CUSR162 | X25\$1601 to X25\$1602 | | CUSR161 to CUSR162 | UserOption1601 to UserOption1602 |
| 34 | 2 | CUSR341 to CUSR342 | X25\$3401 to X25\$3402 | | CUSR341 to CUSR342 | UserOption3401 to UserOption3402 |
| 44 | 2 | CUSR441 to CUSR442 | X25\$4401 to X25\$4402 | | CUSR441 to CUSR442 | UserOption4401 to UserOption4402 |
| 64 | 5 | CUSR641 to CUSR645 | X25\$6401 to X25\$6405 | | CUSR641 to CUSR645 | UserOption6401 to UserOption6405 |
| 72 | 5 | CUSR721 to CUSR725 | X25\$7201 to X25\$7205 | | CUSR721 to CUSR725 | UserOption7201 to UserOption7205 |

User Option Input Panels

On stage, recompile, and relink panels that have not been customized, if you select the **Other options** field, four **User Options** panels (CMNUSR01/2/3/4) are displayed in a series.

| CMNUSR01 | | User Options Part 1 | |
|-----------------------------|------------------------------|---------------------|---|
| Command ==> _____ | | | |
| Name: ACPSRCEE | | | + |
| Type: SRC | Language: COBOL2 | | |
| Compile only | IMS DLITxxx entry | | |
| CICS precompile | Drop include stmts | | |
| Easytrieve object | User option 06 | | |
| User option 07 | User option 08 | | |
| User option 09 | User option 10 | | |
| User option 11 | User option 12 | | |
| User option 13 | User option 14 | | |
| User option 15 | User option 16 | | |
| User option 17 | User option 18 | | |
| User option 19 | User option 20 | | |
| Enter "/" to select option | | | |
| / Mixed Case | | | |

| CMNUSR02 | | User Options Part 2 | |
|------------------------------|------------------|---------------------|---|
| Command ==> _____ | | | |
| Name: ACPSRCEE | | | + |
| Type: SRC | Language: COBOL2 | | |
| Additional build parameters: | | | |
| CUSR641 | _____ | | |
| CUSR642 | _____ | | |
| CUSR643 | _____ | | |
| CUSR644 | _____ | | |
| CUSR645 | _____ | | |
| Enter "/" to select option | | | |
| / Mixed Case | | | |

```

CMNUSR03                               User Options Part 3
Command ==> _____

Name: ACPSRCEE                               +
Type: SRC           Language: COBOL2

Additional component attributes:

CUSR011 . . . . _   CUSR081 . . . . _____   CUSR021 . . . . ___
CUSR012 . . . . _   CUSR082 . . . . _____   CUSR022 . . . . ___
CUSR013 . . . . _   CUSR083 . . . . _____   CUSR023 . . . . ___
CUSR014 . . . . _   CUSR084 . . . . _____
CUSR015 . . . . _   CUSR085 . . . . _____

CUSR031 . . . . ___   CUSR041 . . . . _____   CUSR101 . . . . _____
CUSR032 . . . . ___   CUSR042 . . . . _____   CUSR102 . . . . _____
CUSR033 . . . . ___   CUSR043 . . . . _____

Enter "/" to select option
  / Mixed Case

```

```

CMNUSR04                               User Options Part 4
Command ==> _____

Name: ACPSRCEE                               +
Type: SRC           Language: COBOL2

Additional component attributes:

CUSR161 . . . . _____
CUSR162 . . . . _____
CUSR341 . . . . _____
CUSR342 . . . . _____
CUSR441 . . . . _____
CUSR442 . . . . _____

CUSR721 - CUSR725
_____
_____
_____
_____

Enter "/" to select option
  / Mixed Case

```

The panels shown here are the sample CMNUSR01, CMNUSR02, CMNUSR03, and CMNUSR04 panels that are delivered on the ChangeMan ZMF installer. You can customize these panels to display and process the user options fields that you want to use.

User Options and Exits

User options variables are available in two exits.

| Exit | Function |
|----------|---|
| CMNEX025 | Prohibit package freeze or selective refreeze of Source and Load depending on compile parameters, link edit parameters, and user options used in the last stage job. |
| CMNEX026 | Dynamically allocate additional staging libraries based on component type, user options, and other data passed to the exit. Define relationships between library types so that when a component is deleted, components with the same name in related library types are also deleted. Prohibit delete based on library type, member name, or whether component is promoted. Note: Only the first twenty 1-byte user options (panel field names USROP01 to USROP20) are available in this exit. |

Implementing the User Options

Follow the steps in this section to modify user options components delivered in ChangeMan ZMF libraries so that they satisfy your needs for build processing.

Choose User Options Fields

- 1 Plan how you want user options to store information at the component level and to control build processing. Identify how you want user options to control file tailoring to create build job JCL, set build process parameters, manage target library types with CMNEX026, and any other use you can devise.
- 2 Map each user option variable to the purpose it will serve and the values that will be valid. Choose the shortest fields that will accommodate the data that you will store.
See ["User Options Field Names" on page 65](#) for a description of the 57 available fields.
- 3 Mock up one or more prototype Stage User Options panels, and choose field tags that will fit in the available space.

Modify Sample Stage User Options Panels

- 1 Copy sample Stage User Options panels CMNUSR01-04 into your custom panels library from the CMNZMF PANELS library unloaded from the installer.
- 2 Modify the **)BODY** section of the panels to display field tags that identify the data users should enter in each user option field. To enhance usability, group user options that serve a similar purpose under panel subheadings.
- 3 Modify the **)INIT** section to initialize blank fields. Modify the **.ZVARS =** statement to associate panel fields with the appropriate user option field name.
- 4 Modify the **)PROC** section to validate information entered by the user and to perform cross field edits, if required. Set the value of variable USRPAN to the member name of the next Stage User Options panel, and set USRPAN to blank for the last panel in the chain.

- 5 Copy sample help panel CMNHMSC9 to your custom panels library from the CMNZMF PANELS library unloaded from the ZMF installer. Modify the panel to describe the user options on your custom CMNUSR01 panel. Create new help panels for the other Stage User Options panels. Code the help panel member names in the .HELP = statement in the) INIT section of each Stage User Options panel.
- 6 Verify that your customized Stage User Options panels and help panels look like you want when accessed from these panels.
 - CMNCMPH2 Compile and Link Edit Options
 - CMNQRY22 Query Compile and Link Edit Options
 - CMNRCMP1 Recompil Job Information
 - CMNRCMP3 Recompil Job Information
 - CMNRLNK1 Relink Job Information
 - CMNSTG04 Stage Build
 - CMNSTG05 Stage Mass Build Edit

Modify Exits

User options fields in copybooks CMNEX025 and CMNEX026 are available in copybooks CMNEX025 and CMNEX026 respectively.

- 1 Copy the exit program source you want to modify to your custom source library from the CMNZMF ASMSRC library delivered on the ChangeMan ZMF installer.
- 2 Modify exit program logic to use the user options.

See [Chapter 3, "User Exits" on page 39](#) for general instructions for enabling and coding ChangeMan ZMF exit programs.

Modify Build Skeletons

- 1 Copy stage process skeletons to your custom skeleton library from the CMNZMF SKELS library unloaded from the ZMF installer.
- 2 Modify those skeletons to use the user options variables.



NOTE Skeleton CMN\$\$VAR is imbedded in every compile procedure skeleton delivered with ChangeMan ZMF. Use this skeleton to:

- Translate ChangeMan ZMF ISPF variable names into names that are meaningful to you
- Convert a value in a single user option into multiple variables
- Set one or more variables for file tailoring from combinations of user options

User Option Example

The sample CMNUSR01 panel contains five examples of user options that are fully coded in the compile procedure skeletons delivered with ChangeMan ZMF. You can change these examples to use USROP01-USROP05 for any purpose you choose.

This section describes how the first example user option is used to create stage job JCL that creates object but does not link edit the object into a load module.

User Option Panel CMNUSR01

In the **)BODY** section of panel CMNUSR01, the first user option (USROP01) is labeled "Compile only". In the **)PROC** section, a VER command ensures that Y and N are the only acceptable values for data entered in that field otherwise message CMN132 is issued.

Help panel CMNHMSC9 should be updated to describe this first user option and its purpose.

Variable Skeleton CMN\$\$VAR

In skeleton CMN\$\$VAR, near the top, the value of variable USROP01 is tested. If the value is Y, variable COMPNLY is set to Y. If USROP01 is N, COMPNLY is set to N.

Search skeleton CMN\$\$VAR for occurrences of COMPNLY to see how other variables are set for Object processing.

Compile Procedure CMNCOB2

In compile procedure CMNCOB2, variable COMPNLY is tested for value equal to Y and also for not Y (NE Y). Some skeleton code is selected when the value of COMPNLY is Y, and link edit skeletons CMN\$\$SSI and CMN\$\$LNK are only included when COMPNLY is not Y (NE Y).

HLLX exit requirements

This replaces ISPF dialog panel logic prior to display of the first panel. If you change the first user panel names from the defaults in only the ISPF panel and not the HLLX exit, the **UV** command will not work correctly.

Subsequent panels displays ABCUSR02, ABCUSR03 etc are done as they are today by setting USRPAN to the next panel name in the series.

A simple REXX example follows, if the user is ABCD233 then change the default src and non-src names as follows:

```
if userid = "ABCD233" then  
do  
  if userPanel = "CMNUSR01" then  
    do  
      userPanel = "ABCUSR01"  
    end  
  if userPanel = "CMNUSR11" then  
    do  
      userPanel = "ABCUSR11"  
    end  
end
```

The code must include the Boolean "if" logic about the default panel names, otherwise each time a subsequent panel is called into the exit it will be changed to ABCUSR01/11 and not display any more. In essence this is a 'what to do first time through', subsequent panels are displayed as per usual by setting the USRPAN variable on the user panel

E.G. without the first time "if" logic, if you set the panels to be ABCUSR01-04, when ABCUSR02 is processed by the exit it will be changed to ABCUSR01 again and the panel loop logic will terminate prematurely.

Release ID Variables

Release ID variables are sets of global-level variables created and set by the global administrator. An application administrator can associate one of these release ID variable sets with an application. These variables are available to file tailoring for all batch jobs submitted for an application. You can define up to 54 variables under each release ID.



CAUTION! If you define more than 54 release ID variables, only 54 are kept and only 53 are available during processing. The 54th variable, if defined, will have a null value during processing. No warning is issued.

The original intent of release ID variables was to make it easy to change release numbers for system libraries by including release version variables in system data set names used in ChangeMan ZMF skeletons. Customers have found many ways to use release ID variables to add flexibility to their custom skeletons.



CAUTION! Use caution when designing skeleton customization around release ID variables. These variables and values must be entered and maintained by hand in the ChangeMan ZMF instance that you use to manage components in your production environment. As a more reliable alternative, consider coding variables in skeleton CMN\$\$VAR. You can manage a skeleton like CMN\$\$VAR with ChangeMan ZMF, which will guarantee that what you test is what you install in production.

Release ID variables are defined and updated in the Maintain option of the ChangeMan ZMF Skeleton Maintenance facility.

Accessing Maintain Release ID Variables

Display the **Maintain Release ID Variables** panel using one of these two methods.

- Access the **Skeleton Maintenance Options** panel directly by typing **=A.G.S.M** and pressing **Enter**,
or
- Follow these steps to access the **Maintain Release ID Variables** panel using ChangeMan ZMF menus:
 - a On the **Primary Option Menu**, select **option A Admin**.
 - b On the **Administration Options** menu, select **option G Global**.

- c On the **Global Administration Options** menu, select option **S Skeletons**. The **Skeleton Maintenance Options** menu (CMN3DSKL) is displayed.

```

CMN3DSKL                Skeleton Maintenance Options
Option ==> _____

M Maintain              Maintain skeleton release variables
A Assist                File tailoring assistance of skeleton procedures

```

- d On the **Skeleton Maintenance Options** menu, select option **M Maintain**.

The **Maintain Release ID Variables** panel (CMN3DSM0) is displayed.

```

CMN3DSM0                Maintain Release Id Variables
Option ==> _____

      blank Display release id list          D Delete release id
      R Rename release id                   S Select release id

Release id . . . . . ____ (Blank for list; required for options S,R,D)
New release id . . . ____ (If option R selected)

Enter "/" to select option
  / Confirm delete (if option D selected)

```

This table describes the fields on the **Maintain Release ID Variables** panel.

| Field | Description |
|----------------|--|
| Option | Type one of the following options, or leave the Option line blank and type a line command next to a component name. S Select the release ID typed in the Release ID field. R Rename the release ID typed in the Release ID field using the new name typed in the New Release ID field. D Delete the release ID typed in the Release ID field. blank Display the Release ID List panel where you use line commands to take action against existing release IDs. C Cancel entries on this panel and return to the previous panel. (Long form: CANCEL) |
| RELEASE ID | Type a 1-4 character release ID when Option S, R, or D is selected. |
| NEW RELEASE ID | Type a new name for a release ID when Option R is selected. |
| CONFIRM DELETE | Type Y or N to determine whether a confirmation panel is displayed before ChangeMan ZMF deletes a release ID. Y Display the Confirm Delete panel before a release ID is physically deleted. N Delete a release ID without displaying the Confirm Delete panel. |

Creating a New Release ID

Follow these steps to create a new release ID and new release ID variables.

- 1 On the **Maintain Release Id Variables** panel (CMN3DSM0):
 - a Type **S** in the **Option** line.
 - b Type a 1-4 character name in the **Release id** field. The name cannot start with a number.

```

CMN3DSM0                Maintain Release Id Variables
Option ==> s

      blank Display release id list          D Delete release id
      R Rename release id                   S Select release id

Release id . . . . v8r1 (Blank for list; required for options S,R,D)
New release id . . . (If option R selected)

Enter "/" to select option
/ Confirm delete (if option D selected)
    
```

- 2 Press **Enter** to display the **releaseID - Skeleton Variables** panel (CMN3DSM2).

```

CMN3DSM2                V8R1 - Skeleton Variables                Row 1 to 21 of 21
Command ==> _____ Scroll ==> CSR

Release id description: Test Release number 1

      Variable Description                Value
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
***** Bottom of data *****
    
```

This table describes the fields on the **releaseID - Skeleton Variables** panel.

| Field | Description |
|------------------------|--|
| Command | Type one of the following commands with the appropriate parameter, or leave the command line blank and type a line command next to a release ID variable. LOCATE Locate a variable. (Abbreviation: L) <i>variable</i> SAVE Save changes on this panel. (Abbreviation: S) COPY Copy an existing release ID variable list to this list. (Abbreviation: CO) <i>releaseID</i> REFRESH Refresh the variable list from the package master. (Abbreviation: R) CANCEL Cancel updates on this panel. (Abbreviation: CA) |
| Release id description | Type a description for this release ID. |
| Line Command | Type one of the following line commands. I Insert a line R Repeat a line D Delete a line |
| Variable | Type a variable name to be used in ISPF skeletons. Do not precede the name with ampersand (&) unless you intend for the ampersand to be part of the variable name. |
| Description | Type a description of the variable. |
| Value | Type a 1-32 character value for the variable. If you need a longer variable value, such as a long data set name, you can string two variables together in skeleton code. |

- 3 On the **releaseID - Skeleton Variables** panel, type a **Release id description** and type information in the **Variable**, **Description**, and **Value** fields for each new variable.
- 4 Press **PF3** to save the new release ID.

Maintaining an Existing Release ID

Follow these steps to maintain an existing release ID.

- 1 On the **Maintain Release Id Variables** panel, press **Enter** to display the **Release Id List** panel (CMN3DSM1).

| Id | New | Changed | User | Description | Request |
|----------------------------|-----|------------|---------------|-----------------------|---------|
| _V8R1 | ___ | 2015/02/24 | 00:58 USER015 | Test Release number 1 | |
| ***** Bottom of data ***** | | | | | |

This table describes fields on the **Release Id List** panel.

| Field | Description |
|--------------|--|
| Command | Type one of the following commands with appropriate parameters, or leave the command line blank and type a line command next to a release ID. LOCATE <i>releaseID</i> Locate the specified release ID. (Abbreviation: L) SELECT <i>releaseID</i> Select the specified release ID for edit. (Abbreviation: S) CLEAR <i>releaseID</i> Clear the previous request for the specified release ID. (Abbreviation: CL) DELETE <i>releaseID</i> Delete the specified release ID. REFRESH Refresh the Release Id List panel from the package master. (Abbreviation: REF) RENAME <i>oldreleaseID</i> <i>newreleaseID</i> Rename the specified old release Id to a specified new name (Abbreviation: REN) CANCEL Cancel updates on this panel. (Abbreviation: CA) |
| Line Command | Type one of the following line commands. S Select a release ID for edit. R Rename the release ID to the new name specified in the New field. D Delete the release ID. C Clear the request made for this release ID. |
| Id | Displays the release ID |
| New | Type a new release ID name when the R line command or the RENAME command is used. |
| Changed | Displays the last date and time the release ID was changed. |
| User | Displays the TSO ID of the person who created the release ID or changed it last. |
| Description | Displays the release ID description. |
| Request | Displays the request made on this panel that will be processed when you exit from the panel. If you cancel these requests are not processed. |

- 2 Using commands with parameters typed in the **Command** line, or using **line commands** types on a release ID row, specify the action required for each release ID and the press **Enter**. The requested action appears in the **Request** field on the **Release ID List** panel. Clear a request with the **CLEAR** command or the C line command.
- 3 When you are satisfied with the information in the Request column, press PF3 to exit the panel and execute the requests.

If you requested a **Select** for a release ID, the **releaseID - Skeleton Variables** panel is displayed when you exit. This table describes the fields on the **releaseID - Skeleton Variables** panel.

| Field | Description |
|------------------------|--|
| Command | Type one of the following commands with the appropriate parameter, or leave the command line blank and type a line command next to a release ID variable. LOCATE Locate a variable. (Abbreviation: L) <i>variable</i> SAVE Save changes on this panel. (Abbreviation: S) COPY Copy an existing release ID variable list to this <i>releaseID</i> list. (Abbreviation: CO) REFRESH Refresh the variable list from the package master. (Abbreviation: R) CANCEL Cancel updates on this panel. (Abbreviation: CA) |
| Release id description | Type a description for this release ID. |
| Line Command | Type one of the following line commands. I Insert a line R Repeat a line D Delete a line |
| Variable | Type a variable name to be used in ISPF skeletons. Do not precede the name with ampersand (&) unless you intend for the ampersand to be part of the variable name. |
| Description | Type a description of the variable. |
| Value | Type a 1-32 character value for the variable. If you need a longer variable value, such as a long data set name, you can string two variables together in skeleton code. |

- 4 On the **releaseID - Skeleton Variables** panel, make the additions and changes you want, then press **PF3** to save the updated release ID values.

Associating a Release ID with an Application

To make release ID variables available for file tailoring, you must associate the release ID with an application by specifying the release ID in Application Administration parameters.

Follow these steps to add a release ID to an application.

- 1 Display the **application Parameters - Part 1 of 3** panel by typing **=A.A.1** and pressing **Enter**. As an alternative, you can use the ChangeMan ZMF menu hierarchy to reach the panel:
 - a On the **Administration Options** panel, select Option **A Application**.
 - b On the **Update Application Administration Options** panel, input the application you want to modify, and select Option **1 Parm**s.

The **application Parameters - Part 1 of 4** panel (CMNGLP01) is displayed.

```

CMNGLP01          ACTP Parameters - Part 1 of 4
Command ===> _____

Application description . . . . ACTP Accounts Payable (Base ZMF)
Skeleton release id . . . . . _____ (* for list)
Normal business hours: from . . . 0001
                          to . . . 2300
Audit level . . . . . 4 (0,1,2,3,4,5)
Checkout enforcement rule . . . 1 (1,2,3)
Entity check if rule 2 . . . . . _____
Staging restriction level . . . 1 (1,2,3)
Entity check if level 2 . . . . . _____
Promotion/demotion rule . . . . 0 (0,1,2,3,4)
Cmnaudrc entity check . . . . . _____
Audit package lock . . . . . ALWAYS (Always/Never/Optional)

Enter "/" to select option
- Keep baseline by site
/ Allow temporary packages
- Disallow concurrent checkout
- Allow checkout to personal lib
- Overlay prior staged module
- Validate version during staging

```

- 2 Type a **release ID** in the **Optional Skeleton Release ID** field, or use * for a list to select from.
- 3 Repeatedly press **Enter** to cycle through all **application Parameters** panels until you are returned to the **Update Application Administration Options** panel.

Custom V01-V10 Variables

Custom variables V01 through V10 were added to ChangeMan ZMF early in its development so you can pass information from ISPF panels to skeleton file tailoring when file tailoring is performed in an address space that is different from your ChangeMan ZMF session.

Originally all file tailoring could be performed in your own ISPF ChangeMan ZMF session address space, but you could choose to have file tailoring performed in a separate address space for some functions so that your terminal can be released for other work. This choice is displayed on ChangeMan ZMF ISPF panels as an option for Batch processing.

In ChangeMan ZMF version 5 and later, file tailoring for some functions, such as building install JCL, is always performed in a separate address space. However, you can still use the Batch option for freeze to pass custom V01-V10 variable information to the skeleton file tailoring that builds install JCL.

Data in custom V01-V10 variables is not stored on the package master or the component master. You customize ChangeMan ZMF ISPF panels to add new fields, you assign values to custom V01-V10 variables, and you customize ChangeMan ZMF skeletons to use the custom V01-V10 variables to meet your needs.

Custom V01-V10 Field Names

Custom V01-V10 Variables have the same name on ISPF panels and in skeleton file tailoring.

There are a total of ten Custom V01-V10 fields. They are either 8 or 72 characters long.

| Field Length | Count | Field Names on ISPF Panels | CMNEXITS Field Names (Package Master) | File Tailoring Variable Names |
|--------------|-------|----------------------------|---------------------------------------|-------------------------------|
| 8 | 5 | V01, V02, V03, V04, V05 | Not applicable | Same as ISPF panels |
| 72 | 5 | V06, V07, V08, V09, V10 | Not applicable | Same as ISPF panels |

Using Custom V01-V10 Variables

When you assign a value to a V01-V10 variable in the panel that initiates the following functions, you can use that V01-V10 variable in logic in the skeleton that is file tailored.

- Batch checkout from baseline/promotion
- Batch checkout from package component list (copied forward at create package)
- Staging Versions panel imbedded in batch checkout
- Batch stage from development
- Batch mass stage
- Batch selective mass stage
- Batch recompile
- Batch mass recompile
- Batch selective mass recompile
- Relink
- Full and selective promote
- Full and selective demote

The **Save Previous Version** panel (CMNCMP03) displayed in batch checkout is a special case. This panel is not always displayed for batch checkout, so if it is not displayed, V01-V10 retain the values assigned on the **Checkout** panel (CMNCKOT1). If the **Save Previous Version** panel (CMNCMP03) is displayed, values assigned to V01-V10 replace values set on the **Checkout** panel (CMNCKOT1).

If you attempt to use variable names V01-V10 to pass values from ChangeMan ZMF panels to file tailoring performed in your ChangeMan ZMF session address space, your variable values might not be available to file tailoring. ChangeMan ZMF programs that initiate batch file tailoring clear the value of V01-V10 in the ChangeMan ZMF ISPF session before processing any panel information.

You cannot use values assigned to custom variables V01-V10 in file tailoring for any of the following functions:

- Freeze from Package List (F1 and F2 on CMNLIST3)

- Freeze Package List Options (F1 and F2 on CMNLIST5)
- Online freeze
- Batch freeze

Summary

The following tables summarizes the data that is presented in this chapter.

| User Data Facility | Description | Data Entry | Data Stored |
|--------------------------|---|---|------------------|
| Package User Information | <ul style="list-style-type: none"> ■ 71 fields per package ■ 1-72 character fields | 2 ISPF package information panels provided for your customization | Package record |
| Staging User Options | <ul style="list-style-type: none"> ■ 57 fields per component ■ 1-72 character fields | 4 ISPF component staging panel provided for your customization * | Component record |
| Release ID Variables | <ul style="list-style-type: none"> ■ 54 fields per release ID ■ 1 release ID per application ■ 32 characters in each field | Global Administration ISPF panel | Global record |
| Custom V01-V10 Variables | <ul style="list-style-type: none"> ■ 10 fields ■ 8 and 72 character fields | You add to standard ISPF panels for batch processes | Not stored |

Chapter 5

Utilities

This chapter describes utility programs used in ChangeMan ZMF batch processes. Use the information provided to modify the behavior of these utilities to provide custom ChangeMan ZMF functions that fit your requirements.

| | |
|--|-----|
| CMNBAHST - Initial History Record | 82 |
| CMNBAT90 - Register Build Output Modules | 83 |
| CMNBKRST - VSAM MASTER UNLOAD, RECOVER, LOAD | 94 |
| CMNCICS1 - CICS NEWCOPY | 98 |
| CMNCICS6 - CICS CSD Extract | 106 |
| CMNFXMN - Generate SETSSI Data | 112 |
| CMNIALD0 - Impact Analysis DB2 Load | 115 |
| CMNPML0D - Master File XML Extractor | 117 |
| CMNSSIDN - LINK EDIT Control Preparation | 123 |
| CMNUPDAT - Stacked Reverse Delta Management | 129 |
| CMNWRITE - Copy And Include Management | 135 |
| SERCOPY - Copy Utility | 144 |
| SERPRINT - SYSOUT Compression Facility | 149 |



IMPORTANT! The utilities described in this chapter are stand-alone programs that run in batch jobs under z/OS with JCL. These utilities are not written to be called by other programs, REXX execs, or other macro language routines or scripts.

CMNBAHST - Initial History Record

Utility program CMNBAHST adds an initial component history record for a component in a baseline library that has never been checked out to a package and processed through a change package life cycle. This program also removes history for specified components.

Use this utility is when you first bring a library of components under ChangeMan ZMF control by adding the library as a baseline library in application administration. With CMNBAHST, you can set component information like language, compile procedure, compile parameters, binder options, and user options for members in the new baseline without checking out and staging the components.

You cannot add component history for a component that already has component history.

Program CMNBAHST connects to the ChangeMan ZMF instance specified in the SUBSYS execution parameter, and it updates the component master coded in the started procedure. When you run CMNBAHST, the ChangeMan ZMF instance must be running.

CMNBAHST Input

- Execution parameters in the program PARM statement
- Keyword parameters in the SYSIN DD statement
- Baseline libraries specified in application administration
- Component master specified in the started procedure JCL

Output

Updated component master

Sample JCL

Sample JCL is delivered in member CMNBAHST in the delivered CMNZMF CNTL library.

DD Statements

This table describes DD statements for program CMNBAHST.

| DDNAME | I/O | Purpose |
|----------|-------|--|
| SER#PARM | Input | PDS(E) library containing information used to connect to the ChangeMan ZMF server through TCP/IP. This library must contain a member named #SERx, where x is the one-character subsystem ID of the ChangeMan ZMF instance. |
| SYSIN | Input | File containing 80-byte keyword parameter records. |

PARM Options

The PARM parameter is required in the EXEC statement for CMMBAHST. This table describes CMMBAHST options that are input through the PARM parameter.

| Parameter | Use | Description |
|-----------|----------|--|
| SUBSYS= | Required | Specifies the one-character subsystem ID of the ChangeMan ZMF instance. |
| USER= | Required | Userid of the person or entity that executes CMNBAHST. A userid is required for CMNBAHST to connect to ChangeMan ZMF server programs. This userid is not used to determine security authorization. |

SYSIN Parameters

Keyword parameters for program CMNBAHST that are input to SYSIN are described in comments in job CMNBAHST delivered in the CMNZMF CNTL library.

Additional notes:

- Records with * in the first position are considered comments.
- Blank records are skipped.
- Library types, languages, and compile procedures specified in keyword parameters are validated against definitions in global administration. If the validation fails, processing for the component is skipped.

Return Codes and Error Messages

Return codes for program CMNBAHST are described in comments in job CMNBAHST delivered in the CMNZMF CNTL library.

Program messages are written to SYSPRINT for step HISTORY.

Reporting

Program results are written to SYSPRINT for step HISTORY.

CMNBAT90 - Register Build Output Modules

CMNBAT90 creates transactions for CMNBATCH with information about build output PDS components. CMNBATCH processes these transactions to register the generated components in the package master. CMNBAT90 is included in build procedures for stage, recompile, and relink.



NOTE CMNBAT90 is not used in build procedures for components stored in HFS directories.

If a build output PDS component is a true load module created by the binder (link edit), CMNBAT90 collects additional information for CMNBATCH transactions:

- Subprogram-to-load relationships imbedded in statically linked composite load modules
- Information about statically linked subprograms that were not created in this build process

CMNBAT90 has two methods for analyzing the structure of a composite load module:

- Scan the SYSPRINT listing from the binder - This is the default method and the most accurate. The binder must be executed with the MAP and LIST options to generate binder listing information that is required for CMNBAT90.
- Scan the members in the load library output from the binder. - This is a legacy method, which works well in many cases. However, the binder listing scan method was added because of exceptions discussed in ["CMNBAT90 Notes" on page 89](#).

Unless otherwise indicated by the BINDLIST= execution parameter, CMNBAT90 attempts to use the binder listing scan to analyze the structure of load modules. If this method is not successful, CMNBAT90 automatically changes to the load library scan. Either method can be forced with the BINDLIST= execution parameter.

CMNBAT90 Input

- Program execution parameters
- Keyword SYSIN statements
- Binder listing
- Load library concatenation matching the SYSLIB concatenation in the binder step
- Library containing build output components

Output

- Transactions for program CMNBATCH
- List of the input keyword statements
- Program messages

Sample JCL

This build job JCL fragment, which was file tailored from skeleton CMN\$\$PDB, shows a CMNBAT90 step that creates CMNBATCH transactions to register a DBRM in the package master.

```
//BT90DBR EXEC PGM=CMNBAT90, *** RECORD DBR NAMES
//          COND=(4,LT),
//          PARM='BINDLIST=XLMOD'
//SYSPRINT DD DISP=(,PASS),DSN=&&LIST21D1,
//          UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
//          DCB=(RECFM=FBM,LRECL=133,BLKSIZE=0)
//BAT90IN  DD DISP=(OLD,PASS),DSN=&&DBRMLIB
//BAT90OUT DD DISP=(MOD,PASS),DSN=&&BAT90CTL,
//          UNIT=SYSDA,SPACE=(CYL,(2,1)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
```

```
//SYSIN DD *
PKG=ACTP000041
SLT=SRC
SNM=ACPSRCD1
SID=USER015
SSI=67B2BC9B
LNG=COBOL2
PRC=CMNCOB2
LLT=DBR
SUP=YES
//SYSUDUMP DD SYSOUT=*
//ABNLIGNR DD DUMMY
```

This build job JCL fragment, which was file tailored from skeleton CMN\$\$LNK, shows a CMNBAT90 step that creates CMNBATCH transactions to register a composite load module in the package master.

```
//BT90LOD EXEC PGM=CMNBAT90, *** RECORD LOD NAMES
// COND=(4,LT)
//SYSPRINT DD DISP=(,PASS),DSN=&&LIST51L1,
// UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
// DCB=(RECFM=FBM,LRECL=133,BLKSIZE=0)
//BAT90IN DD DISP=(OLD,PASS),DSN=&&LOAD
//BAT90OUT DD DISP=(MOD,PASS),DSN=&&BAT90CTL,
// UNIT=SYSDA,SPACE=(CYL,(2,1)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//BAT90LST DD DISP=(OLD,PASS),DSN=&&LIST50L1
//BAT90WRK DD DISP=(,DELETE),DSN=&&BAT90WRK,
// UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=0)
//*)IM CMN$$SYL
//BAT90LIB DD DISP=SHR,DSN=CMNTP.S6.ACTP.STG6.#000038.LOD
// DD DISP=SHR,DSN=CMNTP.S6.V810.PROM.S6P1IT.LOD
// DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.OBJ
// DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.LOS
// DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.LOD
// DD DISP=SHR,DSN=CEE.SCEEKED
//*)IM CMN$$SYL END
/* ADDITIONAL SYSIN CONTROL CARDS BELOW COME FROM IMBED OF CMN$$ILC
//SYSIN DD *
PKG=ACTP000038
SLT=LOS
SNM=ACPSRS00
SID=USER015
SSI=67BCF0C2
LNG=COBOL2
PRC=CMNCOB2
RLK=YES
SUP=NO
LLT=LOD
SLB=ACTPLODCMNTP.S6.ACTP.STG6.#000038.LOD
SLB=ACTPLODCMNTP.S6.V810.PROM.S6P1IT.LOD
SLB=ACTPOBJCMNTP.S6.V810.BASE.ACTP.OBJ
SLB=ACTPLOSCMNTP.S6.V810.BASE.ACTP.LOS
SLB=ACTPLODCMNTP.S6.V810.BASE.ACTP.LOD
ILB=ACTPLOSCMNTP.S6.V810.BASE.ACTP.LOS
//SYSUDUMP DD SYSOUT=*
//ABNLIGNR DD DUMMY
```

DD Statements

This table describes DD statements for CMNBAT90.

| DDNAME | I/O | Purpose |
|----------|--------|--|
| SYSIN | Input | 80-byte keyword statements with information for the CMNBATCH transactions |
| BAT90IN | Input | Library containing the build output components to be registered in the package master. |
| BAT90WRK | I/O | Temporary CMNBAT90 work data set |
| BAT90LST | Input | SYSPRINT list from the binder step Note: BAT90LST is only required for the binder listing scan method of analyzing load modules. |
| BAT90LIB | Input | Load library concatenation that matches the SYSLIB concatenation in the binder step Note: BAT90LIB is only required for the load module scan method of analyzing load modules. CMNBAT90 searches this library concatenation to determine the library where each statically linked subprogram originated. |
| BAT90OUT | Output | Transaction records for input to CMNBATCH |
| SYSPRINT | Output | List of SYSIN keyword statements, program messages |

Program Execution Parameters

The PARM= statement is not required for program CMNBAT90. This table describes execution parameters that may be used with program CMNBAT90.

| Parameter | Use | Description |
|-----------|----------|--|
| BINDLIST= | Optional | <p>Force the behavior of CMNBAT90 for analyzing build output for statically linked subprograms. Valid values:</p> <p>XLMOD Suppress load module analysis. Use this option when the build output members at DDname BAT90IN are either non-load modules, or load modules that cannot contain statically linked subprograms, like BMS MAP load or IMS MFS load. Abbreviation: X</p> <p>YES Scan the binder listing at the BAT90LST DD statement for statically linked subprograms.</p> <p>NO Scan every load module in the library at the BAT90IN DD statement for statically linked subprograms.</p> <p>If the BINDLIST execution parameter is omitted, then CMNBAT90:</p> <ol style="list-style-type: none"> 1 Assumes that the library at DDname BAT90IN contains load modules that may contain statically linked subprograms. 2 Opens DDname BAT90LST to scan the binder listing for statically linked subprograms. If that fails, then CMNBAT90... 3 Scans load modules in the library at DDname BAT90IN for statically linked subprograms. |

SYSIN Keyword Statements

CMNBAT90 keyword statements are input to the program through the SYSIN ddname. Format rules include:

- Keyword options must start in position 1
- Comment records are designated by * in position 1
- Blank SYSIN records are permitted

This table describes the keyword statements for CMNBAT90.

| Option | Use | Description |
|-----------------|----------|---|
| * in position 1 | Optional | Comment. |
| PKG= | Required | Package name (10 characters) |
| SLT= | Required | Input library type (like-source library type for stage and recompile, like-NCAL library type for relink) |
| SNM= | Required | Input member name (like-source member name for stage and recompile, link edit control member name for relink) |
| SID= | Required | Stage user's userid |
| SSI= | Required | SETSSI for the new load module |
| LNG= | Required | Language name |
| PRC= | Required | Compile procedure |
| RLK= | Optional | RLK=YES indicates that the build is a recompile or relink. Omitting this keyword indicates that the build is a stage job. |
| SUP= | Optional | YES Suppress the component activation messages issued by a stage job. NO Issue a component activation messages from a stage job. Value is always NO for recompile or relink. |
| LLT= | Required | Target load library type |

| Option | Use | Description |
|--------|----------|--|
| SLB= | Required | <p>Binder SYSLIB library data set information. The data for this keyword is a string of made up of the following:</p> <p>Char Data</p> <p>4 Application</p> <p>3 Library type</p> <p>44 Data set name</p> <p>8 ERO origin of this library - Values:</p> <p> Staging Package staging library</p> <p> Baseline Application baseline library</p> <p> Current ERO current release</p> <p> Prior ERO prior release</p> <p>8 ERO Release</p> <p>8 ERO release area</p> <p>Note: SLB keyword statements are created by skeleton CMN\$\$ILC using the same ISPF tables and selection logic as CMN\$\$SYL uses to create the SYSLIB library concatenation for the binder and for CMNBAT90. If you customize CMN\$\$SYL, then you must customize skeleton CMN\$\$ILC in a parallel manner.</p> |
| ILB= | Required | <p>Binder INCLIB library data set information. The data for this keyword is a string of made up of the following:</p> <p>Char Data</p> <p>4 Application</p> <p>3 Library type</p> <p>44 Data set name</p> <p>8 ERO origin of this library - Values:</p> <p> Staging Package staging library</p> <p> Baseline Application baseline library</p> <p> Current ERO current release</p> <p> Prior ERO prior release</p> <p>8 ERO Release</p> <p>8 ERO release area</p> <p>Note: ILB keyword statements are created by skeleton CMN\$\$ILC using the same ISPF tables and selection logic as CMN\$\$ILL uses to create the INCLIB library concatenation for the binder. If you customize CMN\$\$ILL, then you must customize CMN\$\$ILC in a parallel manner.</p> |

Return Codes and Error Messages

This table describes program return codes for CMNBAT90.

| Return Code | Description |
|-------------|--|
| 0 | Successful execution |
| 4 | CMNBAT90 finished, but the scan method was changed or the relationship analysis was incomplete; see messages at the SYSPRINT DD statement. |
| 8 | CMNBAT90 failed; see messages at the SYSPRINT DD statement. |
| 12 | System error; see messages |



NOTE CMNBAT90 always sets RC=4 for build output components that were not created by the binder. For these components, execution parameter BINDLIST=NO suppresses message CMN4574A but not message CMN4575A.

Program messages are documented in the *ChangeMan ZMF Messages* manual.

Reporting

Program CMNBAT90 lists input keyword statements in a report at the SYSPRINT DD statement. This is an example of the report.

```
ChangeMan(R) ZMF      CMNBATCH - 8.1.0 2015/02/24 22:21:07
Attempting to initiate dialog with ChangeMan ZMF subtask
Session established with ChangeMan ZMF subtask
SYSIN: ACTP000038 90 RTP=ILOD
SYSIN: ACTP000038 90 SLT=LOS
SYSIN: ACTP000038 90 SNM=ACPSRS00
SYSIN: ACTP000038 90 SID=USER015
SYSIN: ACTP000038 90 SSI=67BCF0C2
SYSIN: ACTP000038 90 LNG=COBOL2
SYSIN: ACTP000038 90 PRC=CMNCOB2
SYSIN: ACTP000038 90 RLK=Y
SYSIN: ACTP000038 90 LLT=LST
SYSIN: ACTP000038 90 LNM=ACPSRS00
Component ACPSRS00 is in ACTIVE status and the package master
LOAD record has been updated accordingly.          ACTP000038
LOAD COMPONENT ACTIVATED.                        ACTP000038
LOAD COMPONENT ACTIVATION LOGGED.                ACTP000038
HISTORY RECORD has been updated accordingly.      ACTP000038
SYSIN: ACTP000038 90 CID=
END OF DATA ON SYSIN - TERMINATING
Session terminated with ChangeMan ZMF started task
```

CMNBAT90 Notes

- 1 Best results are obtained when CMNBAT90 can scan the binder listing to obtain the name of the library that provided each statically linked subprogram. This is the default behavior of CMNBAT90.
- 2 The binder must be run with options LIST and MAP for the binder listing to display the information needed by CMNBAT90. If binder options LIST and MAP, are not used,

CMNBAT90 displays message CMN4581A in SYSPRINT, the step return code is set to RC=4, and the load module scan is performed instead of the binder listing scan.

- 3 When CMNBAT90 must analyze a load module to obtain information about statically linked subprograms, that information will be incomplete if any of the following are true:
 - The library concatenation at the BAT90LIB DD statement does not exactly match the concatenation at the SYSLIB DD statement in the binder step.
 - INCLUDE link edit control statements are used to statically link application object modules or load modules from libraries that are not in the concatenation at the BAT90LIB DD statement. Example:


```
INCLUDE INCLIB ( subpgm1)
INCLUDE PRODLIB ( subpgm2)
```
 - The contents of multiple load libraries are combined using a skeleton like CMN\$\$XPL to get around the limit of 128 extents in the binder SYSLIB DD statement.
- 4 There must be an SLB= or ILB= SYSIN keyword statement for each ZMF managed library in the link edit SYSLIB and INCLIB concatenations. If you use INCLUDE link edit control statements that reference a different DDname, you must add an ILB= keyword statement for each ZMF managed library in the concatenation under that DDname. See "[SLB and ILB Keyword Statements](#)" below.
- 5 Statically linked PL/I subprograms are discovered more accurately using BINDLIST=YES. Specifically, PL/I subprograms that have not been cycled through ZMF and do not have ChangeMan ZMF format IDR data will be recognized successfully only if you use BINDLIST=YES.
- 6 CMNBAT90 is not used to create CMNBATCH transactions for compressed listings created in build jobs. CMNBATCH transactions for LST components are generated in file tailoring for skeleton CMN\$\$PCP.

SLB and ILB Keyword Statements

To correctly build CMNBATCH transactions for package master relationship records, CMNBAT90 must find the application and library type for each statically linked subprogram that the binder (linkage editor) obtains from a ZMF managed library.

CMNBAT90 uses the following process to get the application and library type for statically linked subprograms.

- 1 Skeleton CMN\$\$ILC constructs SLB= and ILB= keyword statements for input to CMNBAT90. These records contain a library name and the corresponding application and library type. See "[SLB=" on page 88](#) and "[ILB=" on page 88](#) for the record formats.
 - For every ZMF managed library in the link edit SYSLIB concatenation, CMN\$\$ILC constructs an SLB= statement. The CMN\$\$ILC logic that builds SLB= statements is the same as the logic in CMN\$\$SYC that builds the SYSLIB concatenation.
 - For every ZMF managed library in the link edit INCLIB concatenation for relink, CMN\$\$ILC constructs an ILB= statement. See "[INCLIB and CMNSSIDN" on page 127](#) for information about INCLIB. The CMN\$\$ILC logic that builds ILB= statements is the same as the logic in CMN\$\$ILL that builds the INCLIB concatenation.

- 2 CMNBAT90 finds the name of the library that provided each statically linked subprogram to the linkage editor by either:
 - Analyzing the binder listing (preferred), or by...
 - Searching the library concatenation at the BAT90LIB DD statement, which is built by skeleton CMN\$\$SYL and should mirror the SYSLIB concatenation in the link edit step.
- 3 CMNBAT90 uses the name of the library where a statically linked subprogram originated to get the application and library type for the subprogram from SLB= and ILB= information.



IMPORTANT!

- If you customize the concatenation of ZMF managed libraries in the SYSLIB DD statement, either by customizing skeleton CMN\$\$SYL logic or by hard coding ZMF libraries, you must customize CMN\$\$ILC in the same manner.
- If you customize the concatenation of ZMF managed libraries in the INCLIB DD statement for relink, either by customizing skeleton CMN\$\$ILL logic or by hard coding ZMF libraries, you must customize CMN\$\$ILC in the same manner.
- If your link edit control statements refer to other DD names, you must manually code an ILB= statement for each library concatenated at that DD statement. Example:

```
INCLUDE ACTRLIB(ACRSCN00)

//ACTRLIB DD DISP=SHR,DSN=CMNTP.S4.V711.BASE.ACTR.LCN

ILB=ACTRLCNCMNTP.S4.V711.BASE.ACTR.LCN
```

CMNBAT90 Example - Composite Load Module

The example in this section shows input and output for CMNBAT90 from a relink job for composite load module ACPSRC50 that contains statically linked subprograms ACPSRS5A, ACPSRS5B, ACPSRS5C, and ACPSRS00.

Binder and CMNBAT90 JCL

This JCL fragment shows job steps for the binder and for CMNBAT90 that were file tailored from ISPF skeleton CMN\$\$LNK. Notice the temporary files for the binder listing and the library containing the link edited load module that are passed from the binder to CMNBAT90. This JCL fragment also shows matching libraries in:

- Load library concatenation at the SYSLIB DD statement for the binder at the LINK step
- Load library concatenation at the BAT90LIB DD statement for CMNBAT90
- SLB= keyword statements in the SYSIN DD statement for CMNBAT90

In the same way, INCLIB concatenations match ILB keyword statements.

```
//LNK EXEC PGM=IEWL, *** LINK-EDIT COMPONENT ACPSRS00
// COND=(4,LT),
// PARM=('LIST,XREF,MAP,RENT',
// ')
//SYSPRINT DD DISP=(,PASS),DSN=&&LIST50L1,
```

```

//          UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
//          DCB=(RECFM=FBA,LRECL=121,BLKSIZE=0)
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//*)IM CMN$$OBL
//*)IM CMN$$SYL
//SYSLIB DD DISP=SHR,DSN=CMNTP.S6.ACTP.STG6.#000038.LOD
//          DD DISP=SHR,DSN=CMNTP.S6.V810.PROM.S6P1IT.LOD
//          DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.OBJ
//          DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.LOS
//          DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.LOD
//          DD DISP=SHR,DSN=CEE.SCEELKED
//*)IM CMN$$SYL END
//*)IM CMN$$ILL
//INCLIB DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.LOS
//SYSLMOD DD DISP=(OLD,PASS),DSN=&&LOAD
//SYSLIN DD DISP=(OLD,DELETE),DSN=&&NULLIN
//          DD DISP=(OLD,PASS),DSN=&&LCT
//*)IM CMN$$CND
//*
//BT90LOD EXEC PGM=CMNBAT90, *** RECORD LOD NAMES
//          COND=(4,LT)
//SYSPRINT DD DISP=(,PASS),DSN=&&LIST51L1,
//          UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
//          DCB=(RECFM=FBM,LRECL=133,BLKSIZE=0)
//BAT90IN DD DISP=(OLD,PASS),DSN=&&LOAD
//BAT90OUT DD DISP=(MOD,PASS),DSN=&&BAT90CTL,
//          UNIT=SYSDA,SPACE=(CYL,(2,1)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//BAT90LST DD DISP=(OLD,PASS),DSN=&&LIST50L1
//BAT90WRK DD DISP=(,DELETE),DSN=&&BAT90WRK,
//          UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
//          DCB=(RECFM=FBA,LRECL=121,BLKSIZE=0)
//*)IM CMN$$SYL
//BAT90LIB DD DISP=SHR,DSN=CMNTP.S6.ACTP.STG6.#000038.LOD
//          DD DISP=SHR,DSN=CMNTP.S6.V810.PROM.S6P1IT.LOD
//          DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.OBJ
//          DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.LOS
//          DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.LOD
//          DD DISP=SHR,DSN=CEE.SCEELKED
//*)IM CMN$$SYL END
//* ADDITIONAL SYSIN CONTROL CARDS BELOW COME FROM IMBED OF CMN$$ILC
//SYSIN DD *
PKG=ACTP000038
SLT=LOS
SNM=ACPSRS00
SID=USER015
SSI=67BCF0C2
LNG=COBOL2
PRC=CMNCOB2
RLK=YES
SUP=NO
LLT=LOD
SLB=ACTPLODCMNT.P.S6.ACTP.STG6.#000038.LOD
SLB=ACTPLODCMNT.P.S6.V810.PROM.S6P1IT.LOD
SLB=ACTPOBJCMNT.P.S6.V810.BASE.ACTP.OBJ
SLB=ACTPLOSCMNT.P.S6.V810.BASE.ACTP.LOS
SLB=ACTPLODCMNT.P.S6.V810.BASE.ACTP.LOD
ILB=ACTPLOSCMNT.P.S6.V810.BASE.ACTP.LOS
//SYSUDUMP DD SYSOUT=*
//ABNLIGNR DD DUMMY
//*)IM CMN$$CND
//*)IM CMN$$PAS
//CPYLOD EXEC PGM=SERCOPY, *** COPY TO LOD STAGING LIB
//          REGION=3M,
//          COND=(4,LT),
//          PARM=('RETRY,REALLOC',
//          'OUTDSN(CMNTP.S6.ACTP.STG6.#000038.LOD)')

```

```
//SYSPRINT DD DISP=(,PASS),DSN=&&LIST1001,
//          UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
//          DCB=(RECFM=FBM,LRECL=121,BLKSIZE=0)
//SYSUT1   DD DISP=(OLD,DELETE),DSN=&&LOAD
//SYSUDUMP DD SYSOUT=*
//ABNLIGNR DD DUMMY
```

SYSPRINT Report of Keyword Input

This is an extract from the compressed listing for the relink job showing the SYSPRINT report of CMNBAT90 keyword input.

```
*****
* DDNAME: BT90LOD.SYSPRINT *
*****

ChangeMan(R) ZMF      CMNBAT90 - 8.1.0  TUESDAY FEBRUARY 24, 2015  22:21:02
SYSIN: PKG=ACTP000038
SYSIN: SLT=LOS
SYSIN: SNM=ACPSRS00
SYSIN: SID=USER015
SYSIN: SSI=67BCF0C2
SYSIN: LNG=COBOL2
SYSIN: PRC=CMNCOB2
SYSIN: RLK=YES
SYSIN: SUP=NO
SYSIN: LLT=LOD
SYSIN: SLB=ACTPLODCMNTP.S6.ACTP.STG6.#000038.LOD
SYSIN: SLB=ACTPLODCMNTP.S6.V810.PROM.S6P1IT.LOD
SYSIN: SLB=ACTPOBJCMNTP.S6.V810.BASE.ACTP.OBJ
SYSIN: SLB=ACTPLOSCMNTP.S6.V810.BASE.ACTP.LOS
SYSIN: SLB=ACTPLODCMNTP.S6.V810.BASE.ACTP.LOD
SYSIN: ILB=ACTPLOSCMNTP.S6.V810.BASE.ACTP.LOS
CMN5400I - Time of day at end of job: 22:21:03 - Condition Code on exit: 00
```

IEBGENER of Transactions for CMNBATCH

This is a listing of the component registration transactions created by CMNBAT90 for processing by CMNBATCH. Hex values display as blanks here, but notice the CID= transactions that provide information about subprogram load modules that were not created in this relink job. This information is recorded in subprogram-to-load relationship records in the package master.

```
ACTP  !      RTP=ILOD
ACTP  !      SLT=LOS
ACTP  !      SNM=ACPSRC50
ACTP  !      SID=USER239
ACTP  !      SSI=5F4E19A9
ACTP  !      PRC=CMNCOB2
ACTP  !      RLK=YES
ACTP  !      LLT=LOD
ACTP  !      SUP=NO
ACTP  !      LNM=ACPSRC50
ACTP  !      CID=ACPSRS5A 5F0EF2BC ACTP000034 ACTP LOS I
ACTP  !      CID=ACPSRS5B 5F0EEC33 ACTP000031 ACTP LOS I
ACTP  !      CID=ACPSRS5C 5F0EEC32 ACTP000031 ACTP LOS I
ACTP  !      CID=ACPSRS00 5F0EEC28 ACTP000031 ACTP LOS I
ACTP  !      CID=
```

CMNBKRST - VSAM MASTER UNLOAD, RECOVER, LOAD

Utility program CMNBKRST performs the following three functions the ChangeMan ZMF VSAM package master, component master, and long name component master VSAM KSDS files:

- **Unload** (backup) - The records in the VSAM KSDS files are written to separate QSAM files. In each output file, CMNBKRST writes a header record with the date/time that the file is created.
- **Forward Recovery** - The records in the three QSAM unload files are written to initialized VSAM KSDS files. Forward recovery records in a sequential copy of the CMNRECV file are written to the appropriate VSAM file if the record time/date is more recent than the QSAM unload file header date/time.
- **Load** (restore) - The forward recovery function is executed. However, if no forward recovery records are input to CMNBKRST, or if there are no recovery records that are more recent than the date/time in the QSAM unload file headers, then no forward recovery records are applied, and the operation is effectively a VSAM file load.

Program Execution Parameters

The PARM= statement is always required for program CMNBKRST, and it must have a value. This table describes execution parameters that may be used and the program functions invoked by each.

| PARM= | Function | Description |
|---------|------------------|--|
| BACKUP | Unload | Tells CMNBKRST to perform the unload function |
| RESTORE | Forward recovery | Tells CMNBKRST to perform the forward recovery function. |
| | Load | Tells CMNBKRST to perform the forward recovery function. However, if no forward recovery records are input to CMNBKRST, or if there are no recovery records that are more recent than the date/time in the QSAM unload file headers, then no forward recovery records are applied, and the forward recovery is effectively a VSAM file load. |

CMNBKRST Input and Output

This table shows the inputs and outputs for the three functions performed by CMNBKRST.

| PARM= (Function) | Input | Output |
|------------------------|--|---|
| BACKUP (Unload) | <ul style="list-style-type: none"> ■ Program execution parameter ■ VSAM KSDS package master ■ VSAM KSDS component master ■ VSAM KSDS short name component master | <ul style="list-style-type: none"> ■ QSAM unload package master ■ QSAM unload component master ■ QSAM unload long name component master ■ SYSPRINT report |

| PARAM= (Function) | Input | Output |
|---|---|--|
| RESTORE (Forward Recovery) | <ul style="list-style-type: none"> ■ Program execution parameter ■ QSAM forward recovery file ■ QSAM unload package master ■ QSAM unload component master ■ QSAM unload long name component master | <ul style="list-style-type: none"> ■ VSAM KSDS package master ■ VSAM KSDS component master ■ VSAM KSDS short name component master ■ SYSPRINT report |
| RESTORE (Load) | <ul style="list-style-type: none"> ■ Program execution parameter ■ QSAM unload package master ■ QSAM unload component master ■ QSAM unload long name component master | <ul style="list-style-type: none"> ■ VSAM KSDS package master ■ VSAM KSDS component master ■ VSAM KSDS short name component master ■ SYSPRINT report |

Sample JCL

This sample JCL executes the unload (backup) function of CMNBKRST for the package master, the component master, and the long name component master. There is a sample named BACKUP in the distributed CNTL library, step named BACKUP.

```
//BACKUP EXEC PGM=CMNBKRST,   *** BACKUP VSAM MASTER FILES
//          REGION=4M,
//          PARM=BACKUP
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CMNPMAS  DD DISP=SHR,DSN=CMNTP.S6.V810.CMNZMF.CMNPMAS
//CMNCMPNT DD DISP=SHR,DSN=CMNTP.S6.V810.CMNZMF.CMNCMPNT
//CMNCMPNL DD DISP=SHR,DSN=CMNTP.S6.V810.CMNZMF.CMNCMPNL
//CMNPQSAM DD DISP=(,CATLG),
//          DSN=CMNTP.S6.V810.BACKUP.CMNPMAS(+1),
//          UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=VB,LRECL=5000,BLKSIZE=0)
//CMNCSQSM DD DISP=(,CATLG),
//          DSN=CMNTP.S6.V810.BACKUP.CMNCMPNT(+1),
//          UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=VB,LRECL=5000,BLKSIZE=0)
//CMNCLQSM DD DISP=(,CATLG),
//          DSN=CMNTP.S6.V810.BACKUP.CMNCMPNL(+1),
//          UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=VB,LRECL=5000,BLKSIZE=0)
```

This sample JCL executes the forward recovery or load (restore) function of CMNBKRST for the package master, the component master, and the long name component master.

```
//BKRST EXEC PGM=CMNBKRST,   *** RESTORE MASTER FILES FROM LATEST
//          REGION=4M,COND=(4,LT),   BACKUP AND APPLY CHANGES
//          PARM=RESTORE
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CMNPMAS  DD DISP=SHR,DSN=CMNTP.S6.V810.CMNZMF.CMNPMAS
//CMNCMPNT DD DISP=SHR,DSN=CMNTP.S6.V810.CMNZMF.CMNCMPNT
//CMNCMPNL DD DISP=SHR,DSN=CMNTP.S6.V810.CMNZMF.CMNCMPNL
//CMNPQSAM DD DISP=SHR,DSN=CMNTP.S6.V810.BACKUP.CMNPMAS(0)
//CMNCSQSM DD DISP=SHR,DSN=CMNTP.S6.V810.BACKUP.CMNCMPNT(0)
//CMNCLQSM DD DISP=SHR,DSN=CMNTP.S6.V810.BACKUP.CMNCMPNL(0)
//CMNRQSAM DD DISP=(OLD,DELETE),DSN=&&RECV
/* Add additional recovery datasets from CLEARRCV as needed
//SORTSTAT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,10)
```

```
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,10)
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,10)
//SORTWK04 DD UNIT=SYSDA,SPACE=(CYL,10)
```

DD Statements

This table describes DD statements for CMNBKRST.

| DDNAME | I/O | Purpose |
|----------|--------|---|
| CMNPMAS | * | Package master VSAM KSDS |
| CMNCMPNT | * | Component master VSAM KSDS |
| CMNCMPNL | * | Long name component master VSAM KSDS |
| CMNPQSAM | * | Package master QSAM unload file |
| CMNCSQSM | * | Component master QSAM unload file |
| CMNCLQSM | * | Long name component master QSAM unload file |
| CMNRQSAM | Input | QSAM copy of forward recovery CMNRECV VSAM file |
| SYSPRINT | Output | CMNBKRST record count report |

* See ["CMNBKRST Input and Output"](#) on page 94.

SYSIN Keyword Statements

There are no SYSIN keyword parameters for CMNBKRST.

Return Codes, Completion Codes, and Error Messages

When CMNBKRST encounters a problem:

- Diagnostic messages are displayed in the SYSPRINT data set.
- If an error condition would result in a return code of 8 or greater:
 - CMNBKRST forces an abend.
 - The return code is displayed in the USER COMPLETION CODE.
 - This message is displayed on the operator console (WTO): UNACCEPTABLE RETURN CODE - ABEND.



IMPORTANT! Always check messages in SYSOUT for the CMNBKRST job step, especially if CMNBKRST abends.

This table describes program return codes for CMNBKRST.

| Return Code | Description |
|-------------|--|
| 0 | Successful execution |
| 4 | CMNBKRST finished, but an abnormal condition was encountered; see SYSOUT |
| 8-16 | Fatal error; see SYSPRINT. |

CMNBKRST issues no numbered messages, so there are no CMNBKRST messages in the *ChangeMan ZMF Messages* manual.

Reporting

Program CMNBKRST lists record counts in a report at the SYSPRINT DD statement.



NOTE Record counts for QSAM unload files do not include the header created by CMNBKRST.

This is an example of the report when CMNBKRST is run with PARM=BACKUP.

```
CMNBKRST - EXECUTION BEGINS: 2013/06/01 20:05:55
FUNCTION: BACKUP
INPUT PACKAGE MASTERS:                0003007
OUTPUT PACKAGE MASTERS:               0003007
INPUT SHORT NAME COMPONENT MASTERS:   0001134
OUTPUT SHORT NAME COMPONENT MASTERS:  0001134
INPUT LONG NAME COMPONENT MASTERS:    0000000
OUTPUT LONG NAME COMPONENT MASTERS:   0000000
END OF JOB; RC=0000
```

This is an example of the report when CMNBKRST is run with PARM=RESTORE, but no forward recovery records are input.

```
CMNBKRST - EXECUTION BEGINS: 2013/06/01 20:57:29
FUNCTION: RESTORE
PACKAGE MASTER BACKUP TAKEN:          20130601/20402449
SHORT NAME COMPONENT MASTER BACKUP TAKEN: 20130601/20402449
LONG NAME COMPONENT MASTER BACKUP TAKEN: 20130601/20402449
INPUT RECOVERY RECORDS:               0000000
INPUT PACKAGE MASTERS:                0003007
OUTPUT PACKAGE MASTERS:               0003007
INPUT SHORT NAME COMPONENT MASTERS:   0001134
OUTPUT SHORT NAME COMPONENT MASTERS:  0001134
INPUT LONG NAME COMPONENT MASTERS:    0000000
OUTPUT LONG NAME COMPONENT MASTERS:   0000000
END OF JOB; RC=0000
```

This is an example of the report when CMNBKRST is run with PARM=RESTORE, and forward recovery records are input and applied.

```
CMNBKRST - EXECUTION BEGINS: 2013/06/01 20:32:05
FUNCTION: RESTORE
PACKAGE MASTER BACKUP TAKEN:          20130601/19535684
SHORT NAME COMPONENT MASTER BACKUP TAKEN: 20130601/19535684
LONG NAME COMPONENT MASTER BACKUP TAKEN: 20130601/19535684
INPUT RECOVERY RECORDS:               0001169
INPUT PACKAGE MASTERS:                0002993
OUTPUT PACKAGE MASTERS:               0003007
PACKAGE RECOVERY RECORDS USED:        0000073
INPUT SHORT NAME COMPONENT MASTERS:   0001130
OUTPUT SHORT NAME COMPONENT MASTERS:  0001134
SHORT NAME CMPNT RECOVERY RECORDS USED: 0000012
INPUT LONG NAME COMPONENT MASTERS:    0000000
OUTPUT LONG NAME COMPONENT MASTERS:   0000000
LONG NAME CMPNT RECOVERY RECORDS USED: 0000000
END OF JOB; RC=0000
```

CMNBKRST Notes

- 1 There are situations where you can use IDCAMS in place of CMNBKRST to unload and load the three ChangeMan ZMF VSAM KSDS master files. For example, housekeeping jobs to reorganize the VSAM files can use IDCAMS. However, IDCAMS backups cannot be used for forward recovery, so periodic "backups" of the package master, component master, and long name component master should always use CMNBKRST instead of IDCAMS.
- 2 It is important to keep package and component data synchronized, so you should unload, forward recover, and load the three VSAM files as a set.

However, CMNBKRST skips processing for a file if you code DUMMY in the input and output DD statements for the file. This sample JCL unloads only the long name component master VSAM file.

```
//BACKUP EXEC PGM=CMNBKRST,   *** BACKUP VSAM MASTER FILES
//          REGION=4M,
//          PARM=BACKUP
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CMNPMAS  DD DUMMY
//CMNCMPNT DD DUMMY
//CMNCMPNL DD DISP=SHR,DSN=CMNTP.S6.V810.CMNZMF.CMNCMPNL
//CMNPQSAM DD DUMMY
//CMNCSQSM DD DUMMY
//CMNCLQSM DD DISP=(,CATLG),
//          DSN=CMNTP.S6.V810.BACKUP.CMNCMPNL(+1),
//          UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=VB,LRECL=5000,BLKSIZE=0)
```

- 3 When you run CMNBKRST to forward recover a corrupted or lost ZMF VSAM master file, you can input multiple sequential copies of CMNRECV forward recovery files. These files are:
 - Concatenated at DDname CMNRQSAM
 - Input in any order
 - May have overlapping date ranges

CMNCICS1 - CICS NEWCOPY

Utility program CMNCICS1 refreshes an application load module in a CICS region after ChangeMan ZMF has updated that program in a DFHRPL library.

CMNCICS1 is the driver program for CICS newcopy in ChangeMan ZMF. It validates SYSIN input by verifying that the keyword parameters are grouped and sequenced properly, and then it calls subprogram CMNCICS2 to initiate newcopy commands.



NOTE "Newcopy" is used as a generic term that refers to the CICS load module refresh process that includes PHASEIN as well as NEWCOPY. Where sample commands here show subparameter NEWCOPY or NEW, you can substitute PHASEIN or PHA. NEWCOPY is the default where nothing is specified.

Subprogram CMNCICS2 executes newcopy processing in CICS regions that run on the same LPAR as the batch job that executes CMNCICS1. CMNCICS2 calls subprogram

CMNEMTP using the CICS external call interface (ECI) to execute newcopy processing in CICS regions that run on different LPARs in the same SYSPLEX.

CMNCICS1 or CMNEMTP verifies whether the program to be newcopied is present in a specified library in the DFHRPL concatenation unless this function is suppressed by a CMNCICS1 execution parameter.

There are six options for executing CICS newcopy processing:

| Option | Description |
|------------|--|
| SRB | Compatibility: CICS/ESA 4.1 and below |
| | LPAR: The batch job that executes CMNCICS1 must be on the same LPAR as the CICS region where the program is refreshed. |
| | Method: CMNCICS2 schedules SRB to the CICS region to alter the PPT and force a program refresh. |
| | CICS Trans ID: None |
| | Comments: If used with CICS Transaction Server 1.2 and above, CMNCICS1 gives RC=0 but the target CICS program is not refreshed. |
| ESA | Compatibility: CICS/ESA 4.1 and below |
| | LPAR: The batch job that executes CMNCICS1 must be on the same LPAR as the CICS region where the program is refreshed. |
| | Method: CMNCICS2 uses access registers of the CICS address space to alter the PPT and force a program refresh. |
| | CICS Trans ID: None |
| | Comments: If used with CICS Transaction Server 1.2 and above, CMNCICS1 gives RC=0 but the target CICS program is not refreshed. |
| OPR | Compatibility: All CICS releases |
| | LPAR: The batch job that executes CMNCICS1 must be on the same LPAR as the CICS region. |
| | Method: CMNCICS2 issues an operator modify command to execute CICS supplied operator transaction CEMT under the master console terminal control in the target CICS region: F CICS cicsid,'CEMT SET PROG(pgmname) NEW' |
| | CICS Trans ID: CEMT |
| | Comments: CEMT returns many messages to the master console for each NEWCOPY |
| OPS | Compatibility: CICS TS 1.2 and above |
| | LPAR: The batch job that executes CMNCICS1 must be on the same LPAR as the CICS region where the program is refreshed. |
| | Method: CMNCICS2 issues an operator modify command to execute ChangeMan ZMF transaction SEMT that is processed by program CMNEMTP in the target CICS region: F CICS cicsid,'SEMT SET PROG(pgmname) NEW' |
| | CICS Trans ID: SEMT <ul style="list-style-type: none"> ■ Processed by ZMF program CMNEMTP ■ Executes a subset of CEMT commands ■ Executes with the same authority as CEMT ■ Returns only one message to the OS console |

| Option | Description |
|------------|---|
| OPQ | Compatibility: CICS TS 1.2 and above |
| | LPAR: The batch job that executes CMNCICS1 must be on the same LPAR as the CICS region where the program is refreshed. |
| | Method: CMNCICS2 issues an operator modify command to execute ChangeMan ZMF transaction SEMQ that is processed by program CMNEMTP in the target CICS region: F CICS cicsid,'SEMQ SET PROG(pgmname) NEW' |
| | CICS Trans ID: SEMQ <ul style="list-style-type: none"> ■ Processed by ZMF program CMNEMTP ■ Executes a subset of CEMT commands ■ Executes with the same authority as CEMT ■ Suppresses all messages to the OS console |
| XCI | Compatibility: CICS TS 1.2 and above |
| | LPAR: The batch job that executes CMNCICS1 must be on the same SYSPLEX as the CICS region where the program is refreshed, but it may be on a different LPAR. |
| | Method: CMNCICS2 uses the CICS EXCI facility to call program CMNEMTP in the target CICS region. CMNEMTP issues command: EXEC CICS SET PROGRAM(program) NEWCOPY |
| | CICS Trans ID: SEML – This is a dummy transaction ID required for the EXCI interface. |
| | Comments: XCI is the only way to issue a NEWCOPY in a CICS region that is running on a different LPAR than the batch job that executes CMNCICS1. |



IMPORTANT! XCI is the only option where RC=0 indicates that the newcopy was executed successfully. Options OPR, OPS, and OPQ issue operator commands which do not return a status code.

CMNCICS1 Input

- PARM execution parameters
- SYSIN keyword parameter statements

Output

- Altered PPT or MODIFY commands or EXEC CICS SET PROGRAM... commands, all to initiate newcopy.
- Program execution listing
- Program return code

Sample JCL

JCL to execute program CMNCICS1 is file tailored from skeleton CMN\$CNC, which you customize and imbed as needed in skeletons for promotion, demotion, installation, backout, baseline ripple, and reverse baseline ripple.

```

//JOB LIB DD DISP=SHR,DSN=CMNTP.S4.V710.CMNZMF.CUSTOM.LOAD
// DD DISP=SHR,DSN=CMNTP.S4.V710.CMNZMF.LOAD
// DD DISP=SHR,DSN=CMNTP.S4.V710.SERCOMC.CUSTOM.LOAD
// DD DISP=SHR,DSN=CMNTP.S4.V710.SERCOMC.LOAD
// DD DISP=SHR,DSN=SYS2.CICSTS22.CICS.SDFHEXCI *FOR XCI ONLY
...
//CILCNC EXEC PGM=CMNCICS1, *** CICS NEWCOPY FOR CIL
// COND=(4,LT),
// PARM=(XCI)
//SYSPRINT DD DISP=(,PASS),DSN=&&LISTCNC,
// UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
// DCB=(RECFM=FA,LRECL=133,BLKSIZE=133)
//SYSIN DD *
TARGET=CICSC102
DFHRPL=CMNTP.S4.V610.PROD.ACTP.CIL
PROGRAM=GNLICIS10
/*

```

DD Statements

This table describes the DD statements for program CMNCICS1.

| DDNAME | I/O | Purpose |
|----------|--------|--|
| SYSIN | Input | Input file containing 80-byte keyword parameter records |
| SYSPRINT | Output | Report file that displays information from the execution of CMNCICS1 |
| CMNIN | Input | Alternate for DD name SYSIN |
| CMNOUT | Output | Alternate for DD name SYSPRINT |

PARM Options

The PARM parameter is required in the EXEC statement for CMNCICS1.

The subparameters in the PARM statement are positional and are separated by commas..

```

//CILCNC EXEC PGM=CMNCICS1,
// PARM=(option,check,prefix)

```

This table describes CMNCICS1 options that are input through the PARM parameter.

| Parameter | Use | Description |
|-----------|----------|--|
| option | Required | 3 character code for the method used to execute the newcopy function. Valid values: SRB ESA OPR OPS OPQ XCI Options are described in in the table page 99 . |

| Parameter | Use | Description |
|-----------|----------|--|
| check | Optional | <p>Controls whether the library named in the DFHRPL= SYSIN statement is checked to see if it contains the load module named in the PROGRAM= SYSIN statement. Valid values:</p> <p>CHECK Check the RPL library for the presence of the module to be newcopied. This is the default value if this subparameter is not coded.</p> <p>NOCHECK Do not check the RPL library for the presence of the module to be newcopied.</p> <p>DFHRPL verification is performed by CMNCICS1 for options SRB, ESA, OPR, OPS, and OPQ. Verification is performed by CMNEMTP for option XCI.</p> |
| prefix | Optional | <p>Code 3 characters to replace the first 3 characters of ChangeMan ZMF CICS transactions SEMT, SEMQ, and SEML. The resulting transIDs must be valid, and they must be defined in any CICS region where they will execute.</p> <p>Note: Replacement characters may be coded in the PREFIX= keyword parameter for program CMNCICS1 instead of in the PARM statement. See "SYSIN Parameters" below.</p> |

SYSIN Parameters

Keyword parameters are input to CMNCICS1 through the SYSIN ddname.

```
//SYSIN DD *
PREFIX=prefix
TARGET=cicsid, setoption
  DFHRPL=loadlib
    PROGRAM=pgmname
    PROGRAM=pgmname
    PROGRAM=pgmname
```

- Keyword parameters start in positions 1-60.
- Keyword parameters may be indented to show hierarchy and groupings.
- Each SYSIN record should contain only one keyword parameter.
- Blank SYSIN records are permitted.
- Comment records are designated by * in position 1.
- A maximum of 65,535 DFHRPL= parameters may be input for each TARGET= parameter.
- A maximum of 65,535 PROGRAM= parameters may be input for each TARGET= parameter.

This table describes keyword parameters that are input to CMNCICS1 through the SYSIN DD statement.

| Parameter | Use | Description | | | | | | | | | | | | |
|------------------|---|--|--------|---|------------------|---|-------------|---|-----------|--|-----|--|-----|------------------|
| * in Position 1 | Optional | Denotes a comment. | | | | | | | | | | | | |
| PREFIX= | Optional | Code 3 characters to replace the first 3 characters of ChangeMan ZMF CICS transactions SEMT, SEMQ, and SEML. The resulting transIDs must be valid, and they must be defined in any CICS region where they will execute. If multiple PREFIX records are input, only the last one is retained. Note: Replacement characters may be coded in the PARM statement for program CMNCICS1 instead of in this SYSIN keyword statement. See " PARM Options " on page 101. | | | | | | | | | | | | |
| TARGET= | Required | The TARGET keyword parameter can have two arguments separated by a comma: <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; border: none;">cicsid</td> <td style="border: none;">An identifier for the target CICS region, 1-8 characters. The type of identifier depends on the option you are using.</td> </tr> <tr> <td style="border: none;">CICS region name</td> <td style="border: none;">Use the CICS region name if you are using options SRB, ESA, OPR, OPS, or OPQ.</td> </tr> <tr> <td style="border: none;">VTAM APPLID</td> <td style="border: none;">Use the CICS region VTAM APPLID for option XCI.</td> </tr> <tr> <td style="border: none;">setoption</td> <td style="border: none;">The option to be executed by the SET PROGRAM(pgmname) command.</td> </tr> <tr> <td style="border: none;">NEW</td> <td style="border: none;">Execute NEWCOPY. This is the default option.</td> </tr> <tr> <td style="border: none;">PHA</td> <td style="border: none;">Execute PHASEIN.</td> </tr> </table> <p>Examples: TARGET=CICSA,PHA TARGET=CICSA,NEW TARGET=CICSA</p> | cicsid | An identifier for the target CICS region, 1-8 characters. The type of identifier depends on the option you are using. | CICS region name | Use the CICS region name if you are using options SRB, ESA, OPR, OPS, or OPQ. | VTAM APPLID | Use the CICS region VTAM APPLID for option XCI. | setoption | The option to be executed by the SET PROGRAM(pgmname) command. | NEW | Execute NEWCOPY. This is the default option. | PHA | Execute PHASEIN. |
| cicsid | An identifier for the target CICS region, 1-8 characters. The type of identifier depends on the option you are using. | | | | | | | | | | | | | |
| CICS region name | Use the CICS region name if you are using options SRB, ESA, OPR, OPS, or OPQ. | | | | | | | | | | | | | |
| VTAM APPLID | Use the CICS region VTAM APPLID for option XCI. | | | | | | | | | | | | | |
| setoption | The option to be executed by the SET PROGRAM(pgmname) command. | | | | | | | | | | | | | |
| NEW | Execute NEWCOPY. This is the default option. | | | | | | | | | | | | | |
| PHA | Execute PHASEIN. | | | | | | | | | | | | | |
| DFHRPL = | Required | Load library in the CICS region DFHRPL where the CICS program resides. Must be a fully qualified data set name, up to 44 characters. | | | | | | | | | | | | |
| PROGRAM = | Required | Name of the program to newcopy, 1-8 characters. | | | | | | | | | | | | |

Return Codes and Error Messages

Messages issued by ChangeMan ZMF are described in the ChangeMan ZMF Messages book. This section contains additional information that will be helpful in diagnosing problems with the CMNCICS1 newcopy utility..

| Return Code | Description |
|-------------|---|
| 04 | An error was detected before the newcopy method was executed, but the step was allowed to finish with non-fatal return code. See the messages in SYSPRINT. Examples: CMN7210E TARGET CICS not active on system CMN7205E CMNCICS1 Group has no DFHRPL statements at all |
| 08 | An error was detected before the newcopy method was executed, and the step issued a fatal return code. See the messages in SYSPRINT. Examples: CMN7207E CMNCICS1 DFHRPL dataset failed to allocate CMN7206E CMNCICS1 DFHRPL dataset does not exist |
| 20 | Problem allocating or opening the external CICS interface pipe; these types of errors indicate a problem with inter-system communications or the lack of an EXCI connection in the target CICS. |
| 28 | Indicates a problem with the DPL (Distributed Program Link) to program CMNEMTP; these types of problems indicate an error in the installation of the trans IDs or program within the target CICS. |
| 36 | With CICS abend AXFQ, indicates that the transaction profile parameter INBFMH is not set to ALL. See the CICS resource definition examples provided here. |

| Message Number | Description |
|----------------|--|
| CMN7213E | CICS RETURNED ERRORS CICS regions returned errors when doing newcopy Explanation: CMNCICS2 attempted a CICS NEWCOPY/PHASEIN for an application program, but the application program is not found in the target region. Only programs that have been defined in the CICS system definition file (CSD) and installed on the running CICS system are accessible to ZMF newcopy facilities. Solution: Create a definition for the application program in the target CICS region. |
| CMN7214E | EXCI FAILURE Unable to establish EXCI session with target CICS Explanation: CMNCICS2 is unable to allocate or open an EXCI connection to the target CICS region to perform a NEWCOPY/PHASEIN. The NEWCOPY is not performed. Solution: Ensure the target CICS region is running, and that the EXCI connection has been properly installed. |
| CMN7215E | INSTALL ERROR ZMF newcopy support not installed in target CICS Explanation: CMNCICS2 is unable to start the SEML transaction in the target CICS region to perform NEWCOPY/PHASEIN. The NEWCOPY is not performed. Solution: See the ChangeMan ZMF Installation Guide for the steps to install the SEML transaction in the target CICS region. |

Reporting

The SYSPRINT DD statement for CMNCICS1 displays the following information:

- Program name and title.
- Keyword parameter records input to SYSIN.
- Information and error messages.

Example:

```
CMNCICS1          CICS NEW PROGRAM UTILITY

TARGET=CICSC101
DFHRPL=USER.SERENA.CICSLOAD
PROGRAM=ACCT01
                CMN7209E Program ACCT01 was not found in DFHRPL library
```

Notes and Comments

The examples below show the difference in message volume between option OPR and OPS. The SYSLOG output is also visible in the CICS JESMSGLOG.

This is the JCL, and SYSPRINT output from option OPS, and SYSLOG output.

```
JCL:
//USER015B JOB , ,CLASS=A,NOTIFY=&SYSUID,
//          COND=(4,LT),MSGLEVEL=(1,1),MSGCLASS=X,
//          REGION=4M
//JOBLIB   DD DISP=SHR,DSN=CMNTP.CMN810.LOAD
//          DD DISP=SHR,DSN=CMNTP.SER810.LOAD
//*
//LCXCNC  EXEC PGM=CMNCICS1,   *** CICS NEWCOPY
//          PARM=(OPS,CHECK)
//SYSPRINT DD SYSOUT=*
//SYSIN   DD *
          TARGET=CICSC102,PHA
          DFHRPL=USER.SERENA.CICSLOAD
          PROGRAM=CMNEMTP
***** Bottom of Data *****

SYSPRINT:
CMNCICS1          CICS NEW PROGRAM UTILITY

          TARGET=CICSC102,PHA
          DFHRPL=USER.SERENA.CICSLOAD
          PROGRAM=CMNEMTP
***** Bottom of Data *****

SYSLOG:
S0296623 00000080 +SEMT SET PROG(CMNEMTP) PHA
                Set command completed normally
S0296623 00000080 +Resp=0000 Resp2=0000
```

This is the JCL and SYSPRINT output from option OPR, and also the output to the SYSLOG.

```
//USER015C JOB , ,CLASS=A, NOTIFY=&SYSUID,
//          COND=(4,LT),MSGLEVEL=(1,1),MSGCLASS=X,
//          REGION=4M
//*****
//* FROM USER015.JCL.CNTL(CICSJOB1)
//*
//* JOB TO DO A CICS NEWCOPY
//*****
//JOBLIB   DD DISP=SHR,DSN=CMNTP.CMN810.LOAD
//          DD DISP=SHR,DSN=CMNTP.SER810.LOAD
//*
//LCXCNC   EXEC PGM=CMNCICS1,
//          PARM=(OPR,CHECK)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
          TARGET=CICSC102
          DFHRPL=USER.SERENA.CICSLOAD
          PROGRAM=CMNEMTP
***** Bottom of Data *****

CMNCICS1          CICS NEW PROGRAM UTILITY

          TARGET=CICSC102
          DFHRPL=USER.SERENA.CICSLOAD
          PROGRAM=CMNEMTP
***** BOTTOM OF DATA *

SDSF OPERLOG C001      02/25/2015      0W          COLUMNS 41- 120
COMMAND INPUT ==>          SCROLL ==> CSR
S0296623 00000080 + 581
          581 00000080 Program(CMNEMTP)
          581 00000080 Length(0000009664)
          581 00000080 Language(Assembler)
          581 00000080 Progtype(Program)
          581 00000080 Status( Enabled )
          581 00000080 Sharestatus( Private )
          581 00000080 Copystatus( Notrequired )
          581 00000080 Cedefstatus( Cedef )
          581 00000080 Dynamstatus(Notdynamic)
S0296623 00000080 + Rescount(000) 582
          582 00000080 Usecount(0000000002)
          582 00000080 Dataloc(Any)
          582 00000080 Exekey(Uexekey)
          582 00000080 Executionset( Fullapi )
          582 00000080 Concurrency(Quasirent)
          582 00000080 Remotesystem()
          582 00000080 Runtime( Notknown )
          582 00000080 Jvmclass(
          582 00000080 )
S0296623 00000080 + Jvmclass( 583
          583 00000080 )
          583 00000080 Jvmclass(
          583 00000080 )
          583 00000080 Jvmclass(
          583 00000080 )
          583 00000080 Jvmclass(
          583 00000080 )
          583 00000080 Hotpooling( Nothotpool )
          583 00000080 Jvmprofile(DFHJVMPR)
          583 00000080 NORMAL
S0296623 00000080 + RESPONSE: NORMAL TIME: 01.47.49 DATE: 02.25.15 584
          584 00000080 SYSID=C102 APPLID=CICSC102
```

CMNCICS6 - CICS CSD Extract

The CICS CSD interface lets you extract defined table entries in a CICS CSD file, and translate the results into an editable format. This process is controlled by specifying keywords in the SYSIN data stream.

Export Option

Program CMNCICS6 validates the CMNIN file by verifying the keywords (DFHCSD, EXPORT, IGROUP, and OGROUP) are grouped and sequenced properly.

A group consists of one DFHCSD card, one EXPORT card, one or more IGROUP cards (up to 256), and one OGROUP card. You can specify up to 256 groups in a single execution.

- For each DFHCSD/EXPORT keyword pair, CMNCICS6 will issue a GETMAIN for a work area.
- For each DFHCSD card read CMNCICS6 will lock the VSAM data set specified.
- For each EXPORT card read CMNCICS6 will allocate the PDS data set allocated.
- For each IGROUP and the OGROUP card read CMNCICS6 will build an entry in the work area.

Once all of the data has been read and the work areas built, CMNCICS6 will allocate the SORTWORK data sets. For each work area built, CMNCICS6 will open a member in the PDS data set named by the OGROUP specified. SORT is invoked specifying E15 (input) and E35 (output) exits. The E15 (input) SORT exit will read data from the PDS member and translate the GROUP name to that specified in OGROUP. The E35 (output) SORT exit will translate each record into an 'editable' member in the PDS.

A typical job stream to execute CMNCICS6 EXPORT Option follows.

- The DFHCSD keyword must specify a fully qualified cataloged VSAM data set name not exceeding 44 characters in length.
- The EXPORT keyword must specify a fully qualified PDS data set name not exceeding 44 characters in length.
- The IGROUP keyword must specify a valid GROUP entry on the CICS CSD data set specified with the DFHCSD keyword and must not exceed 8 characters in length.
- The OGROUP keyword must not exceed 8 characters in length.

Basic Format of CMNCICS6 Export Control Statement

The following is an example of JCL for export executing CMNCICS6:

```
//CHGMAN6 JOB (account)
//*
//* JCL for EXPORT
//*
//STEPONE EXEC PGM=CMNCICS6
//STEPLIB DD DSN=somnode.SERENA.CMNZMF.VxRxMx.LOAD,DISP=SHR
//          DD DSN=somnode.SERENA.SERCOMC.VxRxMx.LOAD,DISP=SHR
//CMNOUT DD SYSOUT=*
//CMNIN DD *
DFHCSD=VSAM data set name
EXPORT=PDS data set name
IGROUP=input group
IGROUP=input group
IGROUP=input group
.
.
.
OGROUP=output group
DFHCSD=VSAM data set name
EXPORT=PDS data set name
IGROUP=input group
IGROUP=input group
IGROUP=input group
```

This table describes placeholders in the CMNIN parameter statements.

| Subparameters | Description |
|--------------------|---|
| VSAM data set name | Test data set name of the CICS-defined CSD where IGROUP clusters will be read. |
| PDS data set name | Test data set name of the ChangeMan ZMF PDS where the OGROUP members will be written. |
| input group | Name of an application-defined group name residing in the CICS CSD VSAM data set. |
| output group | Name of the production-defined group name for this application |

Import Option

The purpose of this interface is to let you add CICS table entries to a CICS CSD file from an edited format. The input data is usually created by the CSD export process. The import process is controlled by specifying certain keywords in the SYSIN data stream.

Program CMNCICS6 validate SYSIN by verifying the keywords are grouped and sequenced properly. DFHCSD and IMPORT are the valid SYSIN keywords.

A grouping consists of one DFHCSD card, and one IMPORT card. You can specify up to 256 groups in a single execution.

- For each DFHCSD/IMPORT keyword pair, CMNCICS6 will issue a GETMAIN for a work area.
- For each DFHCSD card read CMNCICS6 will lock the VSAM data set specified.
- For each IMPORT card read CMNCICS6 will allocate the PDS data set allocated.

- Once ALL the SYSIN cards are read and the work areas built, CMNCICS6 will allocate the SORTWORK data sets.
- For each work area built, CMNCICS6 will read each member, and for each member invoke SORT, specifying the E15 (input) and E35 (output) exits.
- The E15 (input) SORT exit will read the PDS member and translate the member into a VSAM data record.
- The E35 (output) SORT exit will add or update the CSD file from the VSAM data record.

A typical job stream to execute CMNCICS6 IMPORT Option follows.

- The CMNIN data stream can start in any column. Multiple imports can be specified.
- The DFHCSD keyword must specify a fully qualified cataloged VSAM data set name not exceeding 44 characters in length.
- The IMPORT keyword must specify a fully qualified PDS data set name not exceeding 44 characters in length.

Basic Format of CMNCICS6 Import Control Statement

The following is an example of JCL for export executing CMNCICS6:

```
//CHGMAN6 JOB (account)
//*
//* JCL for IMPORT
//*
//STEPONE EXEC PGM=CMNCICS6
//STEPLIB DD DSN=somnode.SERENA.CMNZMF.VxRxMx.LOAD,DISP=SHR
// DD DSN=somnode.SERENA.SERCOMC.VxRxMx.LOAD,DISP=SHR
//CMNOUT DD SYSOUT=*
//CMNIN DD *
    DFHCSD=VSAM data set name
    IMPORT=PDS data set name
    DFHCSD=VSAM data set name
    IMPORT=PDS data set name
    DFHCSD=VSAM data set name
    IMPORT=PDS data set name
.
.
.
    DFHCSD=VSAM data set name
    IMPORT=PDS data set name
```

This table describes placeholders in the CMNIN parameter statements.

| Term | Description |
|--------------------|---|
| VSAM data set name | is the production DATA SET name of the CICS defined CSD where the clusters will be written. |
| PDS data set name | is the production DATA SET name of the ChangeMan ZMF PDS from which the members will be read. |

CICS Keywords processed by CMNCICS6

This CMNCICS6 assembler code defines the CICS keywords and their default values for PROGRAMS, MAPSETS, TRANSACTIONS and PROFILES:

```

*
*   Definition and defaults for programs
*
PPTDEF  DS    0D
DC CL80'DEFINE Group()'
DC CL80'      PROGram()'
DC CL80'      Language(Cobol)          cobol | assembler | pli | rpg'
DC CL80'      RELoad(No)                no | yes'
DC CL80'      RESident(No)              no | yes'
DC CL80'      RSl(00)                   0 -24 | public'
DC CL80'      Status(Enabled)           enabled | disabled'
DC X'FF'

*
*   Definition and defaults for mapsets
*
MAPDEF  DS    0D
DC CL80'DEFINE Group()'
DC CL80'      Mapset()'
DC CL80'      RSl(00)                    0 -24 | public'
DC CL80'      Status(Enabled)           enabled | disabled'
DC X'FF'

*
*   Definition and defaults for transactions
*
PCTDEF  DS    0D
DC CL80'DEFINE Group()'
DC CL80'      TRAnSACTION()
DC CL80'      PROGram()
DC CL80'      TWasize(00000)            0 - 32767
DC CL80'      PROFile(DFHICST)
DC CL80'      PArtitionset()
DC CL80'      SStatus(Enabled)          enabled | disabled
DC CL80'      PRIMedsize(00000)        0 - 65520
DC CL80'*    REMOTE ATTRIBUTES
DC CL80'      REMOTESystem()
DC CL80'      REMOTEName()
DC CL80'      TRProf()
DC CL80'      Localq()                  no | yes
DC CL80'*    SCHEDULING
DC CL80'      PRIOrity(001)            0 - 255
DC CL80'      TClass(No)               no | 1 - 10
DC CL80'*    ALIASES
DC CL80'      TAskreq()
DC CL80'      Xtranid()
DC CL80'*    RECOVERY
DC CL80'      DTImout(No)               no | 1 - 7000
DC CL80'      Indoubt(Backout)          backout | commit | wait'
DC CL80'      REStArt(No)               no | yes
DC CL80'      SPurge(No)                no | yes
DC CL80'      TPurge(No)                no | yes
DC CL80'      DUmp(Yes)                  yes | no
DC CL80'      TRACe(Yes)                 yes | no
DC CL80'*    SECURITY
DC CL80'      Extsec(No)                 no | yes
DC CL80'      TRANsec(01)                1 - 64
DC CL80'      RSL(00)                    0 - 24 | public
DC CL80'      RSLC(No)                   no | yes | external
DC X'FF'

```

```

*
*   Definition and defaults for profiles
*
PRFDEF  DS   0D
DC CL80' DEFINE Group()
DC CL80'      profile()
DC CL80'      scrnsize(default)      default | alternate
DC CL80'      modename()
DC CL80'      printercomp(no)       no | yes
DC CL80'*    JOURNALLING
DC CL80'      journal(no)            no | 1 - 99
DC CL80'      msgjrn1(no)           no | input | output | inout
DC CL80'*    PROTECTION
DC CL80'      msginteg(no)          no | yes
DC CL80'      onewte(no)            no | yes
DC CL80'      protect(no)           no | yes
DC CL80'*    PROTOCOLS
DC CL80'      dvsuprt(all)          all | nonvtam | vtam
DC CL80'      inbfmh(no)            no | all | dip | eods
DC CL80'      raq(no)               no | yes
DC CL80'      logrec(no)            no | yes
DC CL80'*    RECOVERY
DC CL80'      nepclass(000)         0 - 255
DC CL80'      rtimeout(no)          no | 1 - 7000
DC X'FF'

```

If the functionality of the CMNCICS6 utility meets the current requirements of your installation, you may continue to maintain your existing PROGRAM and TRANSACTION definitions with the CMNCICS6 utility. If your installation requires use of PROGRAM or TRANSACTION definition parameters that are not supported by CMNCICS6 (for example, the DATALOCATION parameter), or if you wish to use ChangeMan/ZMF to control other RDO resource types (for example, TERMINAL definitions) you must use DFHCSDUP.

The DFHCSDUP utility is supplied by IBM as part of CICS. It does not provide the richness of functionality of CMNCICS6 (multiple DFHCSD statements, IMPORT EXPORT statements) but does support all CICS RDO parameters.

Here is the sample JCL to unload existing definitions from the DFHCSD. The primary purpose of this step would be to pull the definitions out of a test CICS, so that they could be maintained in ZMF:

```

//STEP1 EXEC PGM=DFHCSDUP,REGION=0M,
//      PARM='CSD(READWRITE),PAGESIZE(60),NOCOMPAT'
//STEPLIB DD DSN=cicshlq.SDFHLOAD,DISP=SHR
//*****
/** THIS JCL WILL UNLOAD AN EXISTING DFHCSD GROUP IN A FORMAT THAT**
/** WILL ALLOW IT TO BE LOADED TO ANY OTHER DFHCSD LIBRARY      **
//*****
//DFHCSD DD DISP=SHR,DSN=cicshlq.DFHCSD
//SYSPRINT DD SYSOUT=*
//CBDOUT DD SYSOUT=A
//SYSIN DD *
EXTRACT GROUP(XXXXXXX) USERPROGRAM(DFH0CBDC) OBJECTS
/*

```

Here is the sample JCL to upload new definitions from ZMF to the DFHCSD. The input to this step would be the resource definition stored in ZMF. Existing resource definitions created by CMNCICS6 will work without any changes required:

```
//STEP1 EXEC PGM=DFHCSDUP,REGION=0M,
//          PARM='CSD(READWRITE),PAGESIZE(60),NOCOMPAT'
//STEPLIB DD DSN=cicshlq.SDFHLOAD,DISP=SHR
//*****
/** THIS JCL WILL UPLOAD CICS RESOURCE DEFINITIONS TO THE DFHCSD **
//*****
//DFHCSD DD DISP=SHR,DSN=cicshlq.DFHCSD
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
```

CEDA Language Review

OGROUP members of the PDS subject to ChangeMan ZMF control will have identical format to those supplied and documented in the IBM publication *CICS/VS Resource Definition Online*. CMNCICS6 will read these members and create VSAM records for each definition in an identical manner to the online CICS transaction CEDA.

Although multiple CSD files have been used by more prudent installations, the control of a CSD file by ChangeMan ZMF implies that a production CSD file should be kept separate from a testing CSD file and that the testing file can be subject to change using CEDA whereas the production system should have these resource definition online (RDO) transactions disabled. This eliminates the probability of unauthorized changes to the production environment outside of ChangeMan ZMF control.

In the IBM publication, the RDO defined defaults are shown in parentheses. The required keywords are shown in their entirety. Ranges of values are shown with valid limits.

CMNCICS6 assumes that all source has been exported from a valid CSD file. Therefore, any syntactical errors introduced by users of external editors will cause entries to be ignored by CMNCICS6. Default values will be assigned as shown.

For example, the following is the definition for a program. The information here is the same as that used by CEDA to build a PPT entry.

| | |
|-------------------------|-------------------------|
| PROGm(<i>pgmname</i>) | |
| Language(COBOL) | COBOL Assembler P11 |
| RELoad(No) | No Yes |
| RESident(No) | No Yes |
| RS1(00) | 0-64 Public |
| Status(Enabled) | Enabled Disabled |

CMNFI XMN - Generate SETSSI Data

Program CMNFI XMN checks the SETSSI in the IDR record of load modules to ensure that it contain information compatible with ChangeMan ZMF processing. If a SETSSI is incompatible, CMNFI XMN can update it. CMNFI XMN can also ensure that the load library directory contains the correct SETSSI value.

ChangeMan ZMF audit uses the load module SETSSI as a date/time stamp to discover out-of-synch conditions. When a load module is built by ZMF, the SETSSI is set to an

eight-byte alphanumeric representation of a four byte binary number that is the number of seconds between January 1, 1960 and the link date of the load module. To improve the efficiency of ZMF programs that use the SETSSI, it is also stored in the directory of the load library.

If you add a load module or a load library to ChangeMan ZMF, the SETSSI may be blank or it may not contain the same value that ZMF would calculate. This may prevent audit from detecting out-of-synch conditions. CMNFIXMN prepares your load modules and load libraries for management by ChangeMan ZMF.

CMNFIXMN can be run in two modes. When you use the REPORT execution parameter, CMNFIXMN executes the SETSSI check and report its findings for each load module, but no updates are done. After you examine the report, you can run CMNFIXMN with execution parameter EXECUTE to update the load modules and the load library directory.

Input

Load library containing load modules that were not built by ChangeMan ZMF.

Output

Load library where all load module SETSSI are compatible with ChangeMan ZMF processing and the load library directory contains SETSSI values.

Sample JCL

Sample JCL is delivered in member CMNFIXMN in the delivered CMNZMF CNTL library.

DD Statements

This table describes DD statements for CMNFIXMN.

| DDNAME | I/O | Purpose |
|----------|--------|---|
| LOD | I/O | Load library to be updated with SETSSI. |
| SYSPRINT | Output | Report of mock directory updates or actual updates. |

PARM Options

The PARM parameter is required in the EXEC statement for CMNFIXMN. You must use one of the PARM values listed in this table. There is no default value.

| Parameter | Use | Description |
|-----------|-------------|--|
| REPORT | Alternative | Produces a report of current SETSSI and ZMF calculated SETSSI. No updates are performed. |
| EXECUTE | Alternative | Produces a report of current SETSSI and ZMF calculated SETSSI, and where they are different, updates load modules and the load module directory. |

Return Codes and Error Messages

This table lists return codes for CMNFIXMN.

| Return Code | Description |
|-------------|----------------------------|
| 0 | OK. No errors encountered. |
| 4 | Linkage date not found. |

This table lists abend codes for CMNFIXMN.

| Abend Code | Description |
|------------|--|
| U01 | Unable to open //SYSPRINT DD |
| U02 | Unable to open //LOD DD |
| U03 | LOD Library record format not undefined (not allocated as RECFM=U) |
| U04 | Unable to load //LOD DD (probably sequential) |
| U10 | Unable to stow //LOD DD (probably directory full) |

Reporting

This is a sample from a report from CMNFIXMN executed with PARM=REPORT.

```

CMN (MVS-8.1.0) JOB EXECUTION: DATE=WEDNESDAY FEBRUARY 25, 2015 TIME=12:21:06
CMN INVOCATION: PGM(CMNFIXMN - PROCESS SETSSI ROUTER) COMPILE(20141010 11.58)

  PARM='REPORT'
  PARM interpretation: REPORT; mock execution, NO UPDATES!

  LOD library DSNAME=USER015.ISPLLIB

Now a report of a mock update follows; NO UPDATES!

Module      Dir      Module  Module  SSI  Update
Name      Alias-Of Exec SETSSI  SETSSI  Link Date  Type  Dir
-----
CLS                Yes MISSING 61257D9E 2011/08/24 Genned Yes <MOCK UPDATE>
COLOURS           Yes 610A8BE2 610A8BE2 2011/08/04 Actual No
CURPOS            Yes MISSING 61008925 2011/07/27 Genned Yes <MOCK UPDATE>
DELINKI           Yes MISSING 56D592F0 2006/03/01 Genned Yes <MOCK UPDATE>
HELLOW            Yes MISSING 64809EE4 2013/06/06 Genned Yes <MOCK UPDATE>
...
S0C1              Yes MISSING 662ECC1F 2014/04/28 Genned Yes <MOCK UPDATE>
S0C4              Yes MISSING 662ECD14 2014/04/28 Genned Yes <MOCK UPDATE>

  33 Total members
  9 ALIAS notations
  0 Considered Not Executable
  0 Directory SETSSI unusable, replaced by module SETSSI
  23 Directory SETSSI missing, replaced by module SETSSI
  0 Directory SETSSI does not match module, replaced
  1 Directory SETSSI matches module, no action taken

CMNFIX30: PO(pdse) PROGRAM LIBRARY PROCESSOR COMPLETED. RETURN CODE=00
CMNFIXMN: ROUTER PROCESSING COMPLETED. RETURN CODE=00

```

CMNIALD0 - Impact Analysis DB2 Load

Prior to ChangeMan ZMF 6.1, impact analysis data was stored in sequential files or in DB2 tables. Some customers created their own applications to use the DB2 tables.

To allow customers to continue using those applications, program CMNIALD0 extracts baseline unique number (BUN) data and component relationship data from impact analysis data stores introduced in ChangeMan ZMF 6.1. The program formats that data in records ready for load to DB2 tables CMNBUN and CMNBASE.

Execution JCL for program CMNIALD0 is delivered in sample JCL member LDS2DB2 in the CMNZMF CNTL library. This member also includes a DB2 load step for tables CMNBUN and CMNBASE.

The JCL also includes DB2 DDL for an expanded CMNBASE DB2 table with impact analysis relationship data that was added with ChangeMan ZMF 6.1. The extract files created by program CMNIALD0 contain data to populate the new fields.

CMNIALD0 Input

- BUN information extracted from the impact analysis LDS by program CMNIALU0 in the impact analysis unload process
- Relationship information extracted from the impact analysis LDS by program CMNIALU0 in the impact analysis unload process
- Other impact analysis information read from the IALDS impact analysis LDS

Output

- Sequential file of data ready for load to the CMNBUN DB2 table
- Sequential file of data ready for load to the CMNBASE DB2 table or to the extended CMNBASE table.

Sample JCL

Sample JCL to execute program CMNIALD0 is delivered in member LDS2DB2 in the CMNZMF CNTL library.

The sample JCL also includes:

- Job step to execute DB2 procedure DSNUPROC to load DB2 tables CMNBUN and CMNBASE
- Sample LOAD statements for an extended CMNBASE table
- Sample DDL to create the extended CMNBASE table and indexes.

DD Statements

This table describes DD statements for program CMNIALD0.

| DDNAME | I/O | Purpose |
|----------|--------|--|
| CMNIMPCT | Input | Impact analysis LDS |
| BUNSPACE | Input | BUNSPACE sequential file written by program CMNIALU0 in sample JCL member LDSUNLD |
| RELSpace | Input | The RELSPACE sequential file written by program CMNIALU0 in sample JCL member LDSUNLD |
| BUNDB2 | Output | Reformatted BUN records |
| RELDDB2 | Output | Reformatted relationship records |
| CMNPRINT | Output | Report of records read and records reformatted and written. See "Reporting" on page 116. |



CAUTION! Use the BUNSPACE and RELSPACE files created by program CMNIALU0 in sample JCL member LDSUNLD. Do not use files created by program CMNIA000 that is executed in the online impact analysis data extract (skeleton CMN\$IAX) or in sample JCL member IMPACT.

PARM Options

There are no program execution parameters for program CMNIALD0, so there is no PARM parameter in the EXEC statement.

Return Codes and Error Messages

Error messages are written to DD statement CMNPRINT.

Reporting

This is a sample report from program CMNIALD0.

```

20150225 13474033 ChangeMan(R) ZMF      CMNIALD0 - 8.1.0 Impact Analysis Reformat for DB2
20150225 13474033 Phase I begins
20150225 13474035          55 BUN records read
20150225 13474035          55 BUN records reformatted, written
20150225 13474035 Phase I ends
20150225 13474040 Phase II begins
20150225 13474058          96 relation records read
20150225 13474058          96 relation records reformatted, written
20150225 13474058 Phase II ends

```

Notes or Comments

- Read the comments in sample JCL member LDS2DB2 in the CMNZMF CNTL library.
- For input to program CMNIALD0, use the BUNSPACE and RELSPACE files created by program CMNIALU0 in sample JCL member LDSUNLD. Do not use files created by program CMNIA000 that is executed in the online impact analysis data extract (skeleton CMN\$IAX) or in sample JCL member IMPACT.

- The impact analysis LDS is not accessed through the started task, so the started task can be up or down.
- In *job step* LDS2DB2 in sample JCL member LDS2DB2:
 - The STEPLIB concatenation (or the JOBLIB concatenation if you do not use STEPLIB) must be authorized or the DB2 load step will fail.
 - You might need to add SYSUT1 and SORTWKnn data sets. Check your DSNUPROC cataloged procedure.

DDL for CMNBUN and CMNBASE

This is the DDL for DB2 tables CMNBUN and CMNBASE. (DDL for the extended CMNBASE table is included in sample JCL member LDS2DB2.)

```
CREATE TABLE CMNx.CMNBUN
  (INT_BUN          INTEGER          NOT NULL WITH DEFAULT,
   TXT_APPL        CHAR(4)          NOT NULL WITH DEFAULT,
   TXT_LIBTYPE     CHAR(3)          NOT NULL WITH DEFAULT,
   TXT_LIKE        CHAR(3)          NOT NULL WITH DEFAULT,
   TXT_LIKI        CHAR(1)          NOT NULL WITH DEFAULT,
   TXT_DSNAME      CHAR(44)         NOT NULL WITH DEFAULT)
  IN database.cmnspace2;
COMMIT WORK;
CREATE TABLE CMNx.CMNBASE
  (COMPNAME        CHAR(32)         NOT NULL WITH DEFAULT,
   COMPTYPE        INTEGER          NOT NULL WITH DEFAULT,
   RELATION        CHAR(3)          NOT NULL WITH DEFAULT,
   TOWHAT          CHAR(44)         NOT NULL WITH DEFAULT,
   TOWHATYP        INTEGER          NOT NULL WITH DEFAULT)
  IN database.cmnspace1;
COMMIT WORK;
```

CMNPMLOD - Master File XML Extractor

Data extract program CMNPMLOD (previously called SERPMLOD) uses XML Services to provide you with access to package master and component master data while insulating you from changes in how ChangeMan ZMF data is stored. Extracted data is delivered as XML Service replies, with each reply in a separate sequential file.

Program CMNPMLOD can be run against production master file data or against backup data, but the data must be stored in VSAM files. CMNPMLOD does not require ChangeMan ZMF to be running. CMNPMLOD is available only in batch.

Sample JCL for CMNPMLOD is now delivered in member EXEPMLOD in the vendor CMNZMF.CNTL library.

CMNPMLOD Input

- Package master
- Component master
- Long name component master
- XML services MAPDATA file

Output

One or more QSAM files containing master file data in XML format.

Sample JCL

The following is a sample job for a CMNPMLD step that extracts one type of package master data (global compile procedures).

```
//CMNPMLD EXEC PGM=CMNPMLD,REGION=0M
//STEPLIB DD DISP=SHR,DSN=CMNTP.CMN810.LOAD
//          DD DISP=SHR,DSN=CMNTP.SER810.LOAD
//*****
//* ChangeMan ZMF master files
//CMNPMAS DD DISP=SHR,DSN=CMNTP.S6.V810T06.CMNZMF.CMNPMAS
//CMNCMPNT DD DISP=SHR,DSN=CMNTP.S6.V810T06.CMNZMF.CMNCMPNT
//CMNCMPNL DD DISP=SHR,DSN=CMNTP.S6.V810T06.CMNZMF.CMNCMPNL
//*Note: CMNELDSP is obsolete as of ZMF 7.1.3
//CMNELDSP DD DISP=SHR,DSN=somnode.subsys.CMNELCTX
//*****
//* XML DATASPACE BACKUP
//MAPDATA DD DISP=SHR,DSN=CMNTP.SER810.MAPDATA
//*****
//* TRACES AND DUMPS
//SERPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*****
//* GLOBAL XML DOCUMENT FILES
//CMN$GPRM DD DISP=(,CATLG),DSN=USER015.CMN$GPRM.XMLDATA,
//          UNIT=SYSDA,SPACE=(TRK,(10,10),RLSE),
//          DCB=(RECFM=VB,LRECL=4096,BLKSIZE=0)
```

This is an excerpt of what was created in the CMN\$GPRM data set by the JCL above:

```
***** Top of Data *****
<?xml version="1.0"?>
  <service name="PARMS">
    <scope name="GBL">
      <message name="LIST">
        <result>
          <cmnEnvironment>3</cmnEnvironment>
          <enablePanBaseLib>N</enablePanBaseLib>
          <enableLibrBaseLib>N</enableLibrBaseLib>
          <enableLLamBaseLib>N</enableLLamBaseLib>
          <enableOtherBaseLib>N</enableOtherBaseLib>
          <allowStageOverlay>Y</allowStageOverlay>
          <autoScratchLoadMbr>N</autoScratchLoadMbr>
          <enableJes2Spool>Y</enableJes2Spool>
          <disableCalendar>N</disableCalendar>
          <useSerCompress>N</useSerCompress>
          <createCmpWorkRecs>N</createCmpWorkRecs>
          <showUserPanels>Y</showUserPanels>
          <allowOnlyOneApproval>N</allowOnlyOneApproval>
          <keepBaselineBySite>N</keepBaselineBySite>
          ...
          <enableDisplayOrderDbdOverride>N</enableDisplayOrderDbdOverride>
          <enableDisplayOrderPsbOverride>N</enableDisplayOrderPsbOverride>
          <enableDisplayOrderXmlReport>N</enableDisplayOrderXmlReport>
          <enableDisplayOrderApplication>N</enableDisplayOrderApplication>
          <enableDisplayOrder3dSkel>N</enableDisplayOrder3dSkel>
        </result>
      </message>
    </scope>
```

```
</service>
***** Bottom of Data *****
```

DD Statements

This table describes DD statements for program CMNPMLOD.

| DDNAME | I/O | Purpose |
|-----------|--------|---|
| CMNPMAS | Input | Package master VSAM KSDS |
| CMNCMPNT | Input | Component master VSAM KSDS |
| CMNCMPNL | Input | Long name component master VSAM KSDS |
| MAPDATA | Input | MAPDATA sequential file created by the XMLLOAD job that creates the XMLSPACE VSAM LDS that is coded in the started task JCL |
| SERPRINT | Output | Program messages |
| CMN\$ssss | Output | One or more sequential files containing extracted master file data in XML format, where CMN\$ssss is the XML service name |

Extract processing is triggered by the presence of an output DD statement with a ddname that matches an XML service name.

The ddnames you can use to trigger extracts of package master and component master data are described in the following subtopics.

Global Records:

| DD Name | Service | Scope | Message | Description |
|-----------|----------|----------|---------|------------------------|
| CMN\$GPRM | PARMS | GBL | LIST | General Parameters |
| CMN\$GSIT | SITE | GBL | LIST | Global Sites |
| CMN\$GPRC | PROCS | GBL | LIST | Procedure Names |
| CMN\$GLTP | LIBTYPE | GBL | LIST | Library Types |
| CMN\$RESN | REASONS | SERVICE | LIST | Reason Codes |
| CMN\$GLNG | LANGUAGE | GBL | LIST | Language Names |
| CMN\$GICR | IMSCRGN | GBL | LIST | IMS System Information |
| CMN\$GIDO | IMSOVRD | GBL_DBD | LIST | IMS DBD Overrides |
| CMN\$GIPO | IMSOVRD | GBL_PSB | LIST | IMS PSB Overrides |
| CMN\$GDBP | DB2ADMIN | GBL_PHYS | LIST | DB2 Physical Subsystem |
| CMN\$GDBL | DB2ADMIN | GBL_LOGL | LIST | DB2 Logical Subsystem |

Package Records

| DD Name | Service | Scope | Message | Description |
|-----------|---------|----------|---------|----------------------|
| CMN\$PPRM | PACKAGE | GEN_PRMS | LIST | Package Information |
| CMN\$PDSC | PACKAGE | GEN_DESC | LIST | Package Description |
| CMN\$PIMI | PACKAGE | IMP_INST | LIST | Implementation Inst. |

| DD Name | Service | Scope | Message | Description |
|-----------|----------|----------|---------|--------------------------|
| CMN\$PSCD | PACKAGE | SCH_RECS | LIST | Scheduling Information |
| CMN\$PLTP | LIBTYPE | PKG | LIST | Library Types |
| CMN\$PAPR | APPROVER | PKG | LIST | Approval/Reject/Checkoff |
| CMN\$PPPK | PACKAGE | PRT_PKGS | LIST | Participating Packages |
| CMN\$PAAR | PACKAGE | AFF_APLS | LIST | Affected Applications |
| CMN\$PRBR | PACKAGE | REASONS | LIST | Revert/Backout Reasons |
| CMN\$PURC | PACKAGE | USR_RECS | LIST | Package User Records |

Package "I" Records

| DD Name | Service | Scope | Message | Description |
|-----------|----------|----------|---------|---------------------------|
| CMN\$PSCC | CMPONENT | PKG_COMP | LIST | Source components |
| CMN\$PILC | CMPONENT | PKG_LOD | LIST | Load Information |
| CMN\$PUTL | CMPONENT | PKG_UTIL | LIST | Scratch/Rename info. |
| CMN\$PISC | CMPONENT | SRC_INCL | LIST | Source to Includes |
| CMN\$PCUW | CMPONENT | PKG_WRKL | LIST | Component Work List |
| CMN\$PICR | PACKAGE | IMS_CRGN | LIST | IMS Control Region |
| CMN\$PIAS | PACKAGE | IMS_ACB | LIST | IMS ACB statements |
| CMN\$PIDO | IMSOVRD | PKG_DBD | LIST | IMS DBD overrides |
| CMN\$PIPO | IMSOVRD | PKG_PSB | LIST | IMS PSB Overrides |
| CMN\$PLNK | PACKAGE | PKG_LINK | LIST | Linked Packages |
| CMN\$PINC | CMPONENT | LOD_SUBR | LIST | Load to included CSECTs |
| CMN\$PSIT | SITE | PKG | LIST | Site Information |
| CMN\$PPRH | PACKAGE | PRM_HIST | LIST | Package promote history |
| CMN\$PPCH | CMPONENT | PRM_HIST | LIST | Component promote history |

Application Records

| DD Name | Service | Scope | Message | Description |
|-----------|----------|---------|---------|----------------------------|
| CMN\$APRM | PARMS | APL | LIST | Application parameters |
| CMN\$ASIT | SITE | APL | LIST | Sites, Jobcards |
| CMN\$ALTP | LIBTYPE | APL | LIST | Library Types |
| CMN\$ALNG | LANGUAGE | APL | LIST | Language Names |
| CMN\$AAPR | APPROVER | APL | LIST | Approval List |
| CMN\$APRC | PROCS | APL | LIST | Procedure Names |
| CMN\$BASL | BASELIB | SERVICE | LIST | Baseline Libraries |
| CMN\$PRDL | PRODLIB | SERVICE | LIST | Production Libraries |
| CMN\$PRMS | PROMLIB | SITE | LIST | Promotion Site Information |
| CMN\$PRML | PROMLIB | LIBRARY | LIST | Promotion Site Libraries |

ERO Records

| DD Name | Service | Scope | Message | Description |
|-----------|----------|----------|---------|------------------------------|
| CMN\$RLSM | RLSMRLSE | SERVICE | LIST | Release Data |
| CMN\$RGAP | RLSMAPPR | GLOBAL | LIST | Release Global Approvers |
| CMN\$RASC | RLSMAPPR | ASCAPPRV | LIST | Release Area Approvers |
| CMN\$RARE | RLSMAREA | SERVICE | LIST | Release Area Data |
| CMN\$RAPL | RLSMAPPL | SERVICE | LIST | Release Application Data |
| CMN\$RLTP | RLSMLTYP | SERVICE | LIST | Release Library Type |
| CMN\$RAAP | RLSMAPPR | RELEASE | LIST | Release security entity data |
| CMN\$RPRD | RLSMRLSE | PRIOR | LIST | Release Prior Release |
| CMN\$RSYD | RLSMAPPL | SYSLIB | LIST | Release Application Syslib |
| CMN\$RPRM | RLSMAPPL | PROMOTE | LIST | Release Promotion |
| CMN\$RCLK | RLSMAREA | CMP_LOCK | LIST | Release Component Lock |

Component Records

| DD Name | Service | Scope | Message | Description |
|-----------|----------|----------|---------|------------------------------|
| CMN\$GCGD | CMPONENT | GBL_CDSC | LIST | Global Description |
| CMN\$ACGD | CMPONENT | APL_CDSC | LIST | Application Description |
| CMN\$GCSC | CMPONENT | GBL_SECR | LIST | Component Security |
| CMN\$ACSC | CMPONENT | APL_SECR | LIST | Application Security |
| CMN\$CHIS | CMPONENT | HISTORY | LIST | Component History |
| CMN\$GDPC | CMPONENT | GBL_DPRC | LIST | Designated Compile Procs |
| CMN\$ADPC | CMPONENT | APL_DPRC | LIST | Application Designated Procs |

PARM Options

No PARM parameter is required in the EXEC statement for CMNPMLOD.

Return Codes and Error Messages

This table describes return codes for CMNPMLOD.

| Return Code | Description |
|-------------|---|
| 04 | There was a B37 condition for one or more output files, but processing continued for other output DD statements. See JESMSG LG. |

Reporting

This is an example of EXEPMLOD job output at the JESMSGLG DD statement.

```
***** TOP OF DATA *****
JES2 JOB LOG -- SYSTEM C001 -- NODE MP3JES2

19.02.02 J0620680 ---- SUNDAY, 22 MAR 2015 ----
19.02.02 J0620680 IRR010I USERID USER015 IS ASSIGNED TO THIS JOB.
19.02.02 J0620680 ICH70001I USER015 LAST ACCESS AT 18:47:31 ON SUNDAY, MARCH 22, 2015
19.02.02 J0620680 $HASP373 USER015G STARTED - INIT 1 - CLASS A - SYS C001
19.02.04 J0620680 +SER6702I SERNET XML Dsect Cross Reference. Created: 30 Jan 2015 16:48:44
19.02.04 J0620680 +SER6710I CMNPMSEQ processing - 00001402 records read - key JHFS 000000
19.02.04 J0620680 +SER6708I CMN$GPRM Closed - 00000145 records written
19.02.04 J0620680 - --TIMINGS (MINS.)-- -----PAGING COUNTS-----
19.02.04 J0620680 -STEPNAME PROCSTEP RC EXCP CONN TCB SRB CLOCK SERV WORKLOAD PAGE SWAP VIO SWAPS
19.02.04 J0620680 -CMNPMLD 00 391 290 .00 .00 .0 51156 BATCH 0 0 0 0
19.02.04 J0620680 -USER015G ENDED. NAME-PACKAGE CACHE LOAD TOTAL TCB CPU TIME= .00 TOTAL ELAPSED TIME= .0
19.02.04 J0620680 $HASP395 USER015G ENDED

----- JES2 JOB STATISTICS -----
22 MAR 2015 JOB EXECUTION DATE
128 CARDS READ
189 SYSOUT PRINT RECORDS
0 SYSOUT PUNCH RECORDS
11 SYSOUT SPOOL KBYTES
0.04 MINUTES EXECUTION TIME

***** BOTTOM OF DATA *****
```

Sample CMNPMLOD Extract

This is the first 16 records of CMNPMLOD program output into the data set used at the CMN\$GPRC DD statement.

```
<?xml version="1.0"?>
<service name="PROCS">
<scope name="GBL">
<message name="LIST">
<result>
<procName>CMNCOB2</procName>
<language>COBOL2</language>
<procDesc>Stage COBOL2 source</procDesc>
<displayOrderNo>00000</displayOrderNo>
</result>
<result>
<procName>CMN5SQL</procName>
<language>SQL</language>
<procDesc>Translate, compile, and link SQL Stored Proc</procDesc>
<displayOrderNo>00000</displayOrderNo>
</result>
<result>
<procName>CMNMAPGN</procName>
<language>ASM</language>
<procDesc>CICS BMS MAP Gen</procDesc>
<displayOrderNo>00000</displayOrderNo>
</result>
```

Notes or Comments

- Execution parameter REGION=0M is suggested. The storage required to process up to 64 extract files can be substantial.
- DD statements CMNPMAST, CMNCPNT, and CMNCPNL are optional, but you must input either the package master file or the two component masters.
 - If you are extracting data from the package master only, you can omit DD statements CMNCPNT and CMNCPNL.

- If you are extracting data from the component masters only, you can omit DD statement CMNPMAST.
- Use BLSR to reduce EXCP on the VSAM package master and to shorten job runtimes. Specify BLSR in your JCL as follows:


```
//CMNPMAST DD SUBSYS=(BLSR, 'DDNAME=CMNPMALT', 'STRNO=255')
//CMNPMALT DD DISP=SHR, DSN=CMNTP.S6.V810T06.CMNZMF.CMNPMALT * Package Master
```
- Output extract XML files all have DCB attribute LRECL=4096. However, you can use RECFM FB or VB.
 - In FB files, XML statements occupy the complete 4096 record, with a record break on a new result.
 - In VB files, each XML tag is on separate record.
- XML extract file space requirements depend broadly on the type of data being extracted.
 - For global and application data extracts, a few tracks should suffice depending on the number of library types, sites, applications, etc. in your ChangeMan ZMF system.
 - For package data extracts, the space required is proportional to the number of packages in your system. As an example, for PACKAGE PARMS LIST allow 1 cylinder of standard 3390 space per 90 packages on the package master.
- If a B37 abend occurs on an XML extract file, the extract to the affected file is suspended, but extracts to other files continue. The job ends with RC=04.

CMNSSIDN - LINK EDIT Control Preparation

Program CMNSSIDN prepares link edit control statements for the link edit step in stage, recompile, and relink.

If there are no link edit control statements for the object or load being processed, program CMNSSIDN fabricates the necessary statements and passes them to the subsequent link edit step. The fabricated link edit control statement are not kept after the link edit step.

If you provide link edit control statements in a package staging library or in a baseline member, CMNSSIDN processes those control statement, sometimes modifying them, and passes them to the subsequent link edit step. The control statements in the staging or baseline member are not changed.



NOTE File tailoring for skeleton CMN\$LNK may add INCLUDE statements for CICS language interface modules to the link edit control passed from program CMNSSIDN and input the SYSLIN ddname for the linkage editor.

Whether CMNSSIDN modifies existing link edit control statements, or how it fabricates control statements from scratch, is determined by:

- Object created by previous compile or recompile processing.
- CMNSSIDN program execution parameters included in the JCL when skeleton CMNSSIDN is file tailored.

- CMNSSIDN program SYSIN keyword options included in the JCL when skeleton CMNSSIDN is file tailored.

CMNSSIDN Input

- Object code in a sequential file from a compile step.
- Link edit control statements from a dynamically allocated staging or baseline library.
- Optional program execution parameters.
- Keyword options read through SYSIN.

Output

Link edit control statements ready for a link edit step.

Sample JCL

This is a sample job fragment that illustrates what a CMNSSIDN step can look like. This JCL was created from skeleton CMN\$\$\$SI by ISPF file tailoring.

```
//SSIDN EXEC PGM=CMNSSIDN, *** PROCESS LINK-EDIT CONTROL CARDS
// COND=(4,LT)
//SYSPRINT DD DISP=(,PASS),DSN=&&LIST40S1,
// UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
// DCB=(RECFM=FA,LRECL=133,BLKSIZE=0)
//SYSUDUMP DD SYSOUT=*
//ABNLIGNR DD DUMMY
//OBJ DD DUMMY
//LCT DD DISP=(,PASS),DSN=&&LCT,
// UNIT=SYSDA,SPACE=(TRK,(1,5)),
// DCB=(RECFM=F,LRECL=80,BLKSIZE=0)
//STG DD DISP=(,PASS),DSN=&&NULLLCT,
// UNIT=SYSDA,SPACE=(TRK,(1,1,1),RLSE),
// DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=0)
//SYSIN DD *
BAS=CMNTP.S6.V810.BASE.ACTP.LCT
LCT=ACPSRS00
SSI=67BCF0C2
PKG=ACTP000038
RLK=Y
UIL=Y
OPT=CALL
```

DD Statements

This table describes DD statements for CMNSSIDN.

| DDNAME | I/O | Purpose |
|--------|--------|--|
| OBJ | Input | Sequential file containing object code from a compile step. This DD statement is omitted for relink. |
| LCT | Output | Sequential file containing control statements for a linkage edit or binder step. |

| DDNAME | I/O | Purpose |
|----------|--------|---|
| STG | Input | Staging library for link edit control members. |
| SYSIN | Input | File containing 80-byte keyword option records. |
| SYSPRINT | Output | File that displays information from the execution of CMNSSIDN: See "Reporting" on page 141 . |

Program Execution Parameters

The PARM= statement is not required in CMNSSIDN execution JCL. Any parameter that can be input through the program PARM= statement can be input through a SYSIN control statement using the OPT= keyword. If a parameter is input through both the PARM= statement and an OPT= SYSIN control statement, the SYSIN control statement takes precedence.

This table describes program parameters for program CMNSSIDN.

| Parameter | Use | Description |
|-----------|----------|--|
| CALL | Optional | Pass INCLUDE statements found in staging or baseline link edit control member to the output file at ddname LCT. Default: Omitting this parameter and the NCAL parameter is the same as coding CALL. |
| NCAL | Optional | Do not pass INCLUDE statements found in staging or baseline link edit control member to the output file at ddname LCT. Default: Omitting this parameter and the CALL parameter is the same as coding CALL. |
| NAME | Optional | Pass NAME statements found in staging or baseline link edit control member to the output file at ddname LCT. If no NAME statement is found in a stored link edit control member, or if there is no stored member, fabricate a NAME statement and write to ddname LCT. Default: Omitting this parameter and the NONAME parameter is the same as coding NAME. |
| NONAME | Optional | Do not pass NAME statements found in staging or baseline link edit control members to the output file at ddname LCT. Suppress fabrication of NAME statement if no NAME statement is found in a stored link edit control member, or if there is no stored member. Default: Omitting this parameter and the NAME parameter is the same as coding NAME. |
| DLITASM | Optional | Generate these link edit control statements and write to the output LCT file. INCLUDE SYSLIB(DFSLI000) ENTRY DLITASM |
| DLITCBL | Optional | Generate these link edit control statements and write to the output LCT file. INCLUDE SYSLIB(DFSLI000) ENTRY DLITCBL |

| Parameter | Use | Description |
|-----------|----------|--|
| DLITPLI | Optional | Generate these link edit control statements and write to the output LCT file. INCLUDE SYSLIB(DFSLI000) ENTRY DLITPLI |
| OBJECT | Optional | Change object processing as follows: <ul style="list-style-type: none"> ■ Copy object code from ddname OBJ to ddname LCT. ■ Fabricate SETSSI and IDENTIFY statements and write to ddname LCT. ■ Copy any non-object from ddname OBJ to ddname LCT. Non-object in an object deck is usually a NAME link edit control statement generated by the Easytrieve compiler or by the COBOL compiler when the NAME option is used. Use the OBJECT parameter when staging or recompiling Easytrieve source or COBOL source when the NAME option of the COBOL compiler is used. |

SYSIN Control Statements

CMNSSIDN keyword options are input to the program through the SYSIN ddname.

- Keyword options must start in position 1.
- A SYSIN record should contain only one keyword option.
- Blank SYSIN records are not permitted.
- Comment records are designated by * in position 1.

This table describes keyword options input to CMNSSIDN through the SYSIN ddname.

| Option | Use | Description |
|---------------------|----------|---|
| * in Position 1 | Optional | Denotes a comment. |
| BAS= <i>library</i> | Required | Specifies a baseline library for link edit control members. |
| CST=Y | Optional | Forces CMNSSIDN use the first CSECT name in the object ESD for the generated NAME statement written to ddname LCT. Default: The default for this option is CST=N. |
| LCT= <i>member</i> | Required | Specifies a link edit control member to input to program CMNSSIDN. Default: If you omit this control statement, the output load module name will be set to TEMPNAME. |
| NID= <i>csect</i> | Optional | Suppresses generation of an IDENTIFY statement for the specified CSECT. Up to 64 NID= control statements can be input to CMNSSIDN. |

| Option | Use | Description |
|-----------------------|----------|---|
| OPT= <i>parameter</i> | Optional | <p>Inputs program execution parameters through the SYSIN ddname.</p> <p>Program execution parameters are described in "Program Execution Parameters" on page 125. Example: OPT=NCAL</p> <p>Note: Each OPT= keyword specifies only one CMNSSIDN execution parameter, but there can be multiple OPT= keyword option records input to SYSIN. The following are functionally equivalent:</p> <pre>PARM= ' NAME , CALL , OBJECT '</pre> <p>...and...</p> <pre>//SYSIN DD * OPT=NAME OPT=CALL OPT=OBJECT</pre> |
| PKG= <i>packageID</i> | Required | <p>Specifies the package ID of the component being processed. The package ID is part of the ChangeMan ZMF fingerprint that CMNSSIDN creates for the IDENTIFY statement that is fabricated and written to ddname LCT.</p> <p>Default: If you omit this control statement, the package ID in the fingerprint is set to 10 spaces.</p> |
| RLK=Y | Optional | <p>Indicates that there is no object input to CMNSSIDN.</p> <p>Default: The default for this option is RLK=N.</p> |
| RMB= <i>csect</i> | Optional | <p>Indicates that package audit auto resolve has found a CSECT that must be replaced in the composite load module named in LCT=<i>member</i>. If there is no link edit control for LCT=<i>member</i>, then CMNSSIDN fabricates a REPLACE statement for the specified CSECT and writes the statement to ddname LCT.</p> <p>See "INCLIB and CMNSSIDN" on page 127.</p> |
| SSI= <i>hexvalue</i> | Required | <p>Specifies the hexadecimal value in the SETSSI statement written to ddname LCT. This number is also included in the ChangeMan ZMF fingerprint in the IDENTIFY statement fabricated by CMNSSIDN and written to ddname LCT.</p> <p>Default: If you omit this control statement, the SETSSI in the fingerprint is set to zeros.</p> |
| UIL=Y | Optional | <p>Indicates when CMNSSIDN fabricates a REPLACE link edit control statement, it will fabricate an INCLUDE INCLIB(<i>member</i>) rather than INCLUDE SYSLIB(<i>member</i>) and write it to ddname LCT.</p> <p>See "INCLIB and CMNSSIDN" on page 127.</p> <p>Default: The default for this option is UIL=N.</p> |

INCLIB and CMNSSIDN

Skeleton CMN\$\$ILL builds a library concatenation at ddname INCLIB for relink job JCL. INCLIB contains staging, promotion, and baseline libraries for the relink target load library type.

However, ddname INCLIB is not always referenced when a relink is executed. CMNSSIDN fabricates an INCLUDE INCLIB(*member*) statement when:

- 1 A relink job is initiated by package audit auto resolve to replace one or more statically linked subprograms in composite load module *member*.
- 2 There are no link edit control statements for *member* in baseline or staging libraries. (The composite load module *member* was created with the Automatic Call Library facility of the linkage editor when *member* source was staged or recompiled.)

Return Codes and Error Messages

This table describes user abend codes for program CMNSSIDN.

| Code | Cause |
|------------|--|
| S000 U0005 | Unable to open //SYSPRINT. |
| S000 U0006 | Unable to open //SYSIN. |
| S000 U0007 | Unable to open //OBJ when PARM=OBJECT or OPT=OBJECT in //SYSIN is specified. |
| S000 U0008 | Unable to open //LCT. |
| S000 U0009 | Read error on baseline or staging LCT member. |

Reporting

Program CMNSSIDN reports input keyword options, program execution parameters, what link control statement libraries it used, and the output link edit control statements it wrote to output ddname LCT. The report is written to the SYSPRINT ddname.

This is an example of the report.

```

*****
* DDNAME: SSIDN.SYSPRINT
*****

ChangeMan(R) ZMF      CMNSSIDN - 8.1.0  TUESDAY FEBRUARY 24, 2015  20:46:47
  PARM=' '
SYSIN: BAS=CMNTP.S6.V810.BASE.ACTP.LCT
SYSIN: LCT=ACPSRC1A
SYSIN: SSI=67BCDAA5
SYSIN: PKG=ACTP000038
SYSIN: RLK=
SYSIN: UIL=
SYSIN: OPT=CALL
Options compiled from PARM/SYSIN follow:
  NAME      - Allow "NAME" directive.
  CALL      - Allow "INCLUDE" directives.
END OF DATA ON "OBJ" DETECTED
STAGING "LCT" OPENED
STAGING "LCT" MEMBER NOT FOUND
ATTEMPTING TO ALLOCATE BASELINE "LCT"
BASELINE "LCT" ALLOCATED
BASELINE "LCT" OPENED
BASELINE "LCT" MEMBER NOT FOUND
FABRICATING LCT CARDS FROM SCRATCH
      <...+...1...+...2...+...3...+...4...+...5...+...6...+...7.>.
LCT:      SETSSI 67BCDAA5
LCT:      IDENTIFY ACPSRC1A('ACPSRC1A/67BCDAA5/ACTP000038')
LCT:      NAME ACPSRC1A(R)

```


CMNSSIDN Examples

- Stage simple program without link edit control member

| | |
|-------------------|--|
| Input LCT | None |
| Output LCT | FABRICATING LCT CARDS FROM SCRATCH <...+...1...+...2...+...3...+...4...+...5...+ LCT: SETSSI 67BCDAA5 LCT: IDENTIFY ACPSRC1A('ACPSRC1A/67BCDAA5/ACTP000038') LCT: NAME ACPSRC1A(R) |

- Stage simple program with link edit control member

| | |
|-------------------|---|
| Input LCT | <...+...1...+...2...+...3...+...4...+...5...+ LCT: NAME ACPSRS5C(R) |
| Output LCT | PROCESSING MEMBER IN BASELINE "LCT" <...+...1...+...2...+...3...+...4...+...5...+ LCT: SETSSI 53F61C58 LCT: IDENTIFY ACPSRS5C('ACPSRS5C/53F61C58/ACTP000082') LCT: NAME ACPSRS5C(R) |

- Relink composite load using link edit control member

| | |
|-------------------|---|
| Input LCT | <...+...1...+...2...+...3...+...4...+...5...+ LCT: INCLUDE SYSLIB(ACPSRS5A) LCT: INCLUDE SYSLIB(ACPSRS5B) LCT: INCLUDE SYSLIB(ACPSRS5C) LCT: INCLUDE SYSLIB(ACPSRS00) LCT: NAME ACPSRC50(R) |
| Output LCT | PROCESSING MEMBER IN BASELINE "LCT" <...+...1...+...2...+...3...+...4...+...5...+ LCT: INCLUDE SYSLIB(ACPSRS5A) LCT: INCLUDE SYSLIB(ACPSRS5B) LCT: INCLUDE SYSLIB(ACPSRS5C) LCT: INCLUDE SYSLIB(ACPSRS00) LCT: SETSSI 53F7E98A LCT: NAME ACPSRC50(R) |

- Relink initiated by audit auto resolve for composite load without link edit control member (link edited with Automatic Call Library facility)

| | |
|-------------------|--|
| Input LCT | None |
| Output LCT | FABRICATING LCT CARDS FROM SCRATCH <...+...1...+...2...+...3...+...4...+...5...+ LCT: REPLACE ACPSRS1B LCT: INCLUDE INCLIB(ACPSRC1A) LCT: SETSSI 53F7EADF LCT: NAME ACPSRC1A(R) |

CMNUPDAT - Stacked Reverse Delta Management

Program CMNUPDAT manages the current and prior versions of text components that use stacked reverse delta (SRD) storage means in baseline libraries.

Differences between component versions are stored as reverse deltas (SRD) that can be applied to the full current version to create prior versions. The current version of a component is stored in a baseline PDS(E), library, and all deltas for the component are stored in a single PDS(E) member with the same name in a delta library.

Program CMNUPDAT performs four functions:

- **Baseline Ripple** - Compare the staging library member to the baseline library member to create a set of delta records. Add the delta records to the front of the member in the SRD library, effectively rippling prior versions down the stack. Discard the oldest set of delta records if the set exceeds the maximum number of versions. Replace the current baseline member with the staging library member.
- **Reverse Baseline Ripple (Backout)** - Apply the latest delta records in the SRD library member to the current baseline library member to create the prior version. Replace the baseline library member with this prior version. Delete the latest delta records in the SRD library member, effectively reverse rippling the stack of prior versions.
- **Scratch** - Compress the baseline library member into a set of delta records. Add the delta records to the front of the member in the SRD library, effectively rippling prior versions down the stack. Discard the oldest set of delta records if the set exceeds the maximum number of versions. Delete the member from the baseline library.
- **Copy** - Apply the required sets of delta records to the baseline library member to recreate the requested prior version. Copy this prior version to a specified data set. (If the specified data set is a staging library, this is checkout of a prior version.)

Special cases, like baseline ripple for a new component or scratch for a component that has no prior versions execute these same functions while managing empty baseline and/or SRD library members.

CMNUPDAT Input and Output

Input and output for CMNUPDAT depend on the function being performed.

| Function | Input | Output |
|-----------------------------------|---|--|
| Baseline Ripple | Staging library Baseline library Baseline SRD library | Baseline library Baseline SRD library |
| Reverse Baseline Ripple (Backout) | Baseline library Baseline SRD library | Baseline library Baseline SRD library |
| Scratch | Baseline library | Baseline library Baseline SRD library |
| Copy | Baseline library Baseline SRD library | PDS member or staging library. |

Sample JCL

This is a ChangeMan ZMF installation job fragment that shows a CMNUPDAT step for baseline ripple. This JCL was created from skeleton CMN30SRD by ISPF file tailoring.

```
//UPDSRC EXEC PGM=CMNUPDAT,   *** RIPPLE SRC COMPONENTS
//          REGION=4M,
//          COND=(4,LT),
//          PARM='APPLY,REALLOC,MAXLEVEL(9)'
//*)IM CMN$$ENQ
//SYSUT3 DD DISP=(MOD,DELETE),
//          DSN=CMNTP.S6.V810.BASE.ACTP.SRC.ENQ,
//          UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//CMNUPDAT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//ABNLIGNR DD DUMMY
//SYSUDUMP DD SYSOUT=*
//BASELIB DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.SRC
//DELTALIB DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.SRC.DELTA
//STAGELIB DD DISP=SHR,DSN=CMNTP.S6.ACTP.STG6.#000039.SRC
//SYSIN DD *
ACTCOB01
```

DD Statements

This table describes DD statements for program CMNUPDAT.

| DDNAME | I/O | Purpose |
|----------|-----|--|
| SYSUT3 | N/A | Creates an enqueue to single thread jobs |
| SYSUT4 | I/O | Temporary work data set |
| CMNUPDAT | O | Listing from the process that was executed. Examples: <ul style="list-style-type: none"> ■ SERCMPAR listing of text differences that were converted into delta records ■ CMNDELTA listing that shows delta records that were applied to create a prior version |
| SYSPRINT | O | Listing that displays actions taken for each member processed. |
| BASELIB | I/O | Baseline library. Note: This DD name can be changed with the BASELIB(ddname) subparameter of the PARM statement. |
| DELTALIB | I/O | Stacked reverse delta library. Must be RECFM=FB. If incorrect, message CMN5114A - Stacked reverse delta minus baseline must be RECFM=FB Note: This DD name can be changed with the DELTALIB(ddname) subparameter of the PARM statement. |

| DDNAME | I/O | Purpose |
|----------|-----|---|
| STAGELIB | I | Output PDS(E) library for generated member. This DD statement must refer to a PDS(E). If you want to print the generated member, use this library as input to IEBGENER. Note: This DD name can be changed with the STAGELIB(ddname) subparameter of the PARM statement. |
| SYSIN | I | Members to be processed. Multiple member names that are coded on the same SYSIN record are delimited by space or comma. Names may be coded be on multiple SYSIN records. This DD statement is ignored if the MEMBER(mem,...) subparameter of the PARM statement is used. |

PARM Options

The PARM parameter is required in the EXEC statement for CMNUPDAT. This table describes CMNUPDAT options that are input through the PARM parameter.

| Parameter | Use | Description |
|------------------|----------|---|
| Program function | Required | Specifies the function to be performed. Code one of the following: APPLY Baseline ripple RESTORE Reverse baseline ripple (backout) DELETE Scratch COPY Recreate a prior version and copy to another library or data set. These function are described at the top of section "CMNUPDAT - Stacked Reverse Delta Management" on page 129. |
| COMPRESS | Optional | Indicates that the baseline library uses compressed format. |
| RETRY | Optional | If an out-of-space condition occurs in an output PDS library, compress the PDS and retry. |
| REALLOC | Optional | If more space is required for an output library, reallocate the library. |
| ABEND | Optional | Abend program CMNUPDAT if any error occurs. |
| BASELIB(ddname) | Optional | Specifies an alternate ddname for the baseline library. Default: BASELIB |
| DELTALIB(ddname) | Optional | Specifies an alternate ddname for the delta library. Default: DELTALIB |
| STAGELIB(ddname) | Optional | Specifies an alternate ddname for the staging library. Default: STAGELIB |
| ALLMEM | Optional | Process all members. This parameter is ignored if the MEMBER() execution parameter is used or if members are listed in SYSIN. |
| MEMBER(mem,...) | Optional | Specifies the members to be processed. Member names are separated by commas. If this parameter is coded, SYSIN is ignored. If this parameter is omitted, the member names are read from SYSIN. |

| Parameter | Use | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------------|--|--|------------|--|-------|--------------------------------------|-------|--------------------------------------|--------|--|--------|--|-----|--------------------------------------|-----|--|-----|--|--------|--|---|--|-------|--|---------|--|------|--|------|--|-----|--|-----|--|---------|--|------|--|-----|--|
| VERSION(n) | Optional | Specifies the number of prior level deltas to apply for the RESTORE or COPY function. Default: 1 (the level prior to the current baseline version) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| MAXLEVEL(n) | Optional | Sets the maximum number of prior levels saved in the delta member to n. Delta records for level n+1 are discarded when a new set of delta records is added. Default: No limit Example: If application administration specifies 10 levels for a baseline configuration, then n = 9. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| TEXT(a) | Optional | Specifies comparison parameters for creating delta records. Default: TEXT(PANEL) Note: Differences in spaces, and commas in the use of COBOL, are ignored. Precede any of the subparameters below with the \$ character to flag as changed any line where the only difference is in the use of spaces (and commas in the case of COBOL). Examples: TEXT(\$.) or \$COBOL <table border="0"> <tr> <td>. (period)</td> <td>The first four records are analyzed to identify the target language to determine the kind of text compare that should be done.</td> </tr> <tr> <td>COBOL</td> <td>Positions 7 through 72 are compared.</td> </tr> <tr> <td>PANEL</td> <td>Positions 1 through 80 are compared.</td> </tr> <tr> <td>REPORT</td> <td></td> </tr> <tr> <td>SCRIPT</td> <td></td> </tr> <tr> <td>ALC</td> <td>Positions 1 through 72 are compared.</td> </tr> <tr> <td>BAL</td> <td></td> </tr> <tr> <td>JCL</td> <td></td> </tr> <tr> <td>PASCAL</td> <td></td> </tr> <tr> <td>C</td> <td></td> </tr> <tr> <td>CLIST</td> <td></td> </tr> <tr> <td>FORTRAN</td> <td></td> </tr> <tr> <td>PL/1</td> <td></td> </tr> <tr> <td>PL/I</td> <td></td> </tr> <tr> <td>PL1</td> <td></td> </tr> <tr> <td>PLI</td> <td></td> </tr> <tr> <td>NATURAL</td> <td></td> </tr> <tr> <td>REXX</td> <td></td> </tr> <tr> <td>RPG</td> <td></td> </tr> </table> | . (period) | The first four records are analyzed to identify the target language to determine the kind of text compare that should be done. | COBOL | Positions 7 through 72 are compared. | PANEL | Positions 1 through 80 are compared. | REPORT | | SCRIPT | | ALC | Positions 1 through 72 are compared. | BAL | | JCL | | PASCAL | | C | | CLIST | | FORTRAN | | PL/1 | | PL/I | | PL1 | | PLI | | NATURAL | | REXX | | RPG | |
| . (period) | The first four records are analyzed to identify the target language to determine the kind of text compare that should be done. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| COBOL | Positions 7 through 72 are compared. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PANEL | Positions 1 through 80 are compared. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REPORT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SCRIPT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ALC | Positions 1 through 72 are compared. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BAL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| JCL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PASCAL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| C | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CLIST | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FORTRAN | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PL/1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PL/I | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PL1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PLI | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| NATURAL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REXX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| RPG | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Notes or Comments

When stacked reverse delta files are created in the baseline ripple process, special characters and codes are inserted in delta records to tell ChangeMan ZMF how to apply delta records and uncompress text.



CAUTION! If your components include the codes and characters used by program CMNUPDAT, do not use SD-Stacked Reverse Delta for the baseline storage means in application baseline configuration. Use P - Standard PDS instead.

These are the special characters and codes that you should avoid in text managed by the SRD storage means.

| Special Characters and Codes | Notes |
|------------------------------|---|
| <UPDATE> | |
| <ADD> | |
| <DELETE> | |
| <NULL> | |
| <STATS | |
| <*STAMP | |
| <*END.OF.MEMBER> | |
| <*END.OF.DELTA.DECK> | |
| <n, | n is any integer. |
| <n> | n is any integer, and the rest of the record must be blank for this to be detected as ZMF SRD reserved text. This is to avoid conflict with Focus code in the format "<n> text". |
| X'FF03' | <p>This code is reserved for the first byte of SRD members. Members starting with x'FF03' that are not compressed may be incorrectly processed as compressed in the following cases:</p> <ul style="list-style-type: none"> ■ Print and copy baseline components in ChangeMan ZMF ISPF option 1.B. ■ Compare of staging component to promotion or baseline when the library type is LST but the component is not a compressed listing. LST is currently reserved for compressed listings. |

CMNWRITE - Copy And Include Management

CMNWRITE is executed as a step in a ChangeMan ZMF stage (compile or build) job. It parses source code and selectively expands copybooks into the source. The expanded source is written to the file at DD statement SYSOFILE, which is then used as input into a precompiler or compiler.

CMNWRITE attempts to resolve copybook names by searching PDS(E) libraries concatenated at the SYSLIB DD statement. After searching SYSLIB libraries, CMNWRITE searches any CA Panvalet and CA Librarian libraries that are named in SYSIN keyword control statements.

CMNWRITE can process copybooks nested up to 99 levels.

For copybooks that are resolved, including nested copybooks, CMNWRITE writes source-to-copy relationship records to the package master. Audit uses these records to find SYNCH5, SYNCH15, and SYNCH16 errors in package components. These records are also used to create relationship records used by impact analysis.



CAUTION! If you omit CMNWRITE from a custom build process for like-source components that use copybooks, audit will be unable to detect some out-of-synch conditions.

If a copybook name is successfully resolved, but in-line expansion is either not required or not possible, CMNWRITE writes the copybook to a separate PDS at DD statement SYSUT3, which is included at the bottom of the SYSLIB concatenation in the subsequent precompile or compile step.

If a copybook is expanded in the source, CMNWRITE can generate comment box at the top of the expanded code that displays its level of nesting and member information from the library where it was found.

CMNWRITE Input

- Execution parameters in the program PARM statement
- Keyword parameters in the SYSIN DD statement
- Source code in a PDS or sequential file
- PDS libraries to be searched for copybooks
- CA Panvalet libraries to be searched for copybooks
- CA Librarian files to be searched for copybooks

Output

- Source code with copybooks expanded in-line
- Source code comments for every expanded copybook showing ISPF statistics for the copybook, the level of nesting under the source member, and the number of lines in the copybook
- A PDS library containing copybooks that have not been expanded in-line

- SYSPRINT output displaying the library search order and a report of copybooks detected

Sample JCL

This is a fragment from a stage job submitted by ChangeMan ZMF. This JCL was created by file tailoring skeleton CMN\$\$WRT.

```

/(* )IM CMN$$WRT
//WRITE EXEC PGM=CMNWRITE, *** PARSE/EXPAND COMPONENT CTST
// COND=(4,LT),
// PARM=(' SUBSYS=6,USER=USER015',
// ')
/(* )IM CMN$$SPR
//SER#PARM DD DISP=SHR,DSN=CMNTP.SER814.C6.TCPIPORT
//SYSPRINT DD DISP=(,PASS),DSN=&&LIST10W1,
// UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE)
/(* )IM CMN$$SYC
//SYSLIB DD DISP=SHR,DSN=CMNTP.S6.ACTP.STG6.#000081.CPY
// DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.CPY
// DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.CP2
//SYSIFILE DD DISP=(OLD,PASS),DSN=&&SOURCE(CTST)
//SYSOFILE DD DISP=(,PASS),DSN=&&WRITE,
// UNIT=SYSDA,SPACE=(CYL,(1,1)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//SYSUT3 DD DISP=(,PASS),DSN=&&CPYLIB,
// UNIT=SYSDA,SPACE=(CYL,(1,1,20),RLSE),
// DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=0)
//ABNLIGNR DD DUMMY
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
TYP=CPY/CMNTP.S6.ACTP.STG6.#000081.CPY
TYP=CPY/CMNTP.S6.V810.BASE.ACTP.CPY
TYP=CP2/CMNTP.S6.V810.BASE.ACTP.CP2
CMP=CTST.SRC
LNG=COBOL2
PKG=ACTP000081
/(* )IM CMN$$CND
/(* )IM CMN$$CO2

```

DD Statements

This table describes DD statements for CMNWRITE.

| DDNAME | I/O | Purpose |
|----------|-------|--|
| SER#PARM | Input | PDS(E) library containing information used to connect to the ChangeMan ZMF server through TCP/IP. This library must contain a member named #SERx, where x is the one-character subsystem ID of the ChangeMan ZMF instance. |
| SYSIFILE | Input | File containing source code. This DD statement must point to a sequential file or a PDS(E) member. |
| SYSIN | Input | File containing 80-byte keyword control records. |
| SYSLIB | Input | PDS libraries containing copybook members. The library concatenation for this DD statement is usually built by skeleton CMN\$\$SYC. |

| DDNAME | I/O | Purpose |
|----------|--------|---|
| SYSOFILE | Output | Sequential file containing source code with expanded copybooks. The file written by this DD statement is passed to precompile or compile steps. |
| SYSPRINT | Output | Report file that displays information from the execution of CMNWRITE: See "Reporting" on page 141 . |
| SYSUT3 | Output | PDS containing copybook members that could be detected in the input source code but were not expanded in the source written to SYSOFILE. The library created from this DD statement can be included at the bottom of the SYSLIB concatenation for precompile and compile steps. |

CA Panvalet and CA Librarian libraries are specified in SYSIN keyword control statements and are dynamically allocated. See ["SYSIN Parameters" on page 137](#). These libraries are searched after the SYSLIB concatenation is exhausted.

PARM Options

The PARM parameter is required in the EXEC statement for CMNWRITE. This table describes CMNWRITE options that are input through the PARM parameter.

| Parameter | Use | Description |
|-----------|----------|---|
| SUBSYS= | Required | Specifies the one-character subsystem ID of the ChangeMan ZMF instance. |
| USER= | Required | Userid of the person or entity that executes CMNWRITE. A userid is required for CMNWRITE to connect to ChangeMan ZMF server programs. This userid is not used to determine security authorization. |
| EXPAND | Optional | Indicates whether or not to expand copybooks in source code output to SYSOFILE. EXPAND Expand detected copybooks in output source and write unexpanded copybooks to SYSUT3. This is the default value. NOEXPAND Do not expand detected copybooks in output source and write all detected copybooks to SYSUT3. |

SYSIN Parameters

CMNWRITE keyword parameters are input to CMNWRITE through the SYSIN ddname.

- Keyword parameters must start in positions 1-60.
- A SYSIN record should contain only one keyword parameter.
- Blank SYSIN records are permitted.
- Comment records are designated by * in position 1.

This table describes keyword parameters input to CMNWRITE through the SYSIN ddname.

| Parameter | Use | Description |
|-----------------|----------|---|
| * in Position 1 | Optional | Denotes a comment. |
| CMP= | Optional | Specifies a component name and library type. Format: CMP=ccccccc.ttt where ccccccc Component name ttt Library type If the component cannot be found in the component master, the return code is set to at least 4. See "CMNWRITE and Audit" on page 143 . |
| COBSYN= | Optional | Defines a synonym for COBOL or Assembler COPY verb. Multiple synonyms are allowed. Code a separate COBSYN= for each synonym. Note: Copybooks resolved using this parameter are not expanded in-line. These copybooks are written to the PDS at the SYSUT3 DD statement for input to the compile step in the SYSLIB concatenation. |
| EOSPERIOD= | Optional | This parameter was introduced for backward compatibility when the default function of CMNWRITE was modified to require an end-of -sentence period for COBOL COPY statements so that the COPY REPLACING statement could be correctly parsed. OPTIONAL Do not require end-of-sentence periods for COBOL COPY statements. Using this parameter may affect detection of the COPY REPLACING phrase. |
| EXPAND= | Options | Provides finer control over the copy and include structures that CMNWRITE detects and expands. ALL Default - Expand all types of copy PANVALET Only expand CA Panvalet ++INCLUDE LIBRARIAN Only expand CA Librarian -INC COPY Only expand COBOL/Assembler COPY PLI Only expand PL/1 %INCLUDE CEE Only expand C #include SQL Only expand EXEC SQL INCLUDE NONE No expansion (like the NOEXPAND program parameter) Notes: <ul style="list-style-type: none"> ■ Any combination of the above is valid. ■ ALL overrides everything but NONE. ■ NONE overrides everything. ■ PARM=NOEXPAND overrides all EXPAND keyword parameters. |
| FAPIW= | Optional | Specifies a substitution character for FIS Systematics EXEC API PROC member name processing that is enabled by SYSIN parameter SITE=FIDE. |

| Parameter | Use | Description | | | | | | | | | | | | | | | | | | | | |
|-----------|--|---|-------|------------------------------|----|-----------|-----|-----------|---|-------------------------|----|--------------------------|----|------------|------|---------|-----|---------|----|------|-------|--|
| LIB= | Optional | <p>Specifies the DSN of a CA Librarian baseline library to be searched for copybooks.</p> <ul style="list-style-type: none"> ■ Up to ten PAN= and LIB= keyword parameters may be input in any combination. ■ These libraries are searched in the order the LIB= and PAN= records are read in SYSIN. ■ Libraries specified in LIB= and PAN= keyword parameters are searched after the SYSLIB concatenation is exhausted. | | | | | | | | | | | | | | | | | | | | |
| LIBSYN= | Optional | <p>Defines a synonym for CA Librarian -INC verb. Multiple synonyms are allowed. Code a separate LIBSYN= for each synonym.</p> <p>Note: Copybooks resolved using this parameter are not expanded in-line. These copybooks are written to the PDS at the SYSUT3 DD statement for input to the compile step in the SYSLIB concatenation.</p> | | | | | | | | | | | | | | | | | | | | |
| LNG= | Optional | <p>Determines how CMNPARSE analyzes source code to find copy and include statements:</p> <table border="0"> <thead> <tr> <th>Value</th> <th>Source Parsed As Language...</th> </tr> </thead> <tbody> <tr> <td>AS</td> <td>Assembler</td> </tr> <tr> <td>BAL</td> <td>Assembler</td> </tr> <tr> <td>C</td> <td>(C followed by a blank)</td> </tr> <tr> <td>C+</td> <td>(C+ followed by a blank)</td> </tr> <tr> <td>EZ</td> <td>Easytrieve</td> </tr> <tr> <td>FORT</td> <td>FORTRAN</td> </tr> <tr> <td>MFS</td> <td>IMS MFS</td> </tr> <tr> <td>PL</td> <td>PL/I</td> </tr> <tr> <td>blank</td> <td>CMNPARSE attempts to differentiate between Assembler and COBOL</td> </tr> </tbody> </table> <p>Notes:</p> <ul style="list-style-type: none"> ■ If PL, then source will be searched for %INCLUDE. ■ If C, then source will be searched for #INCLUDE. ■ In all languages, source is searched for CA Panvalet ++INCLUDEs and CA Librarian -INC. ■ Any characters coded after those listed above are ignored. However, if the codes above are not complete, the language may be incorrectly identified. For example, CPLUS would be considered COBOL since it is not followed by a blank or plus sign and blank. However, PL2 would still be considered PL/I. | Value | Source Parsed As Language... | AS | Assembler | BAL | Assembler | C | (C followed by a blank) | C+ | (C+ followed by a blank) | EZ | Easytrieve | FORT | FORTRAN | MFS | IMS MFS | PL | PL/I | blank | CMNPARSE attempts to differentiate between Assembler and COBOL |
| Value | Source Parsed As Language... | | | | | | | | | | | | | | | | | | | | | |
| AS | Assembler | | | | | | | | | | | | | | | | | | | | | |
| BAL | Assembler | | | | | | | | | | | | | | | | | | | | | |
| C | (C followed by a blank) | | | | | | | | | | | | | | | | | | | | | |
| C+ | (C+ followed by a blank) | | | | | | | | | | | | | | | | | | | | | |
| EZ | Easytrieve | | | | | | | | | | | | | | | | | | | | | |
| FORT | FORTRAN | | | | | | | | | | | | | | | | | | | | | |
| MFS | IMS MFS | | | | | | | | | | | | | | | | | | | | | |
| PL | PL/I | | | | | | | | | | | | | | | | | | | | | |
| blank | CMNPARSE attempts to differentiate between Assembler and COBOL | | | | | | | | | | | | | | | | | | | | | |

| Parameter | Use | Description |
|-----------|----------|--|
| OPT= | Optional | <p>Output formatting option</p> <p>FLOWER Default - Add a comment "flower box" at the top of each copybook that is expanded in the source and written to SYSOFILE.</p> <p>NOFLOWER Suppress the comment flower box in expanded copybooks.</p> <p>CONTINUE Specifies that existing ISIC records for this component are NOT deleted. Use this subparameter in the second CMNWRITE step of a dual compile scenario where the first CMNWRITE step creates ISIC records and the second CMNWRITE step creates more. Existing ISIC entries are cached. Duplicate entries in the second CMNWRITE step are not added. The existing entries with possibly older copybook data are retained. This may be needed to ensure that certain SYNCH conditions are not missed</p> |
| PAN= | Optional | <p>Specifies the DSN of a CA Panvalet baseline library to be searched for copybooks.</p> <ul style="list-style-type: none"> ■ Up to ten PAN= and LIB= keyword parameters may be input in any combination. ■ These libraries will be searched in the order the LIB= and PAN= records are read in SYSIN. ■ Libraries specified in LIB= and PAN= keyword parameters are searched after the SYSLIB concatenation is exhausted. |
| PKG= | Optional | <p>Specifies the change package ID.</p> <p>If the package cannot be found in the package master, the return code is set to at least 4. See "CMNWRITE and Audit" on page 143.</p> |
| PLISYN= | Optional | <p>Defines a synonym for PL/1 %INCLUDE verb. Multiple synonyms are allowed. Code a separate PLISYN= for each synonym.</p> <p>Note: Copybooks resolved using this parameter are not expanded in-line. These copybooks are written to the PDS at the SYSUT3 DD statement for input to the compile step in the SYSLIB concatenation.</p> |
| PVSYN= | Optional | <p>Defines a synonym for CA Panvalet ++INCLUDE verb. Multiple synonyms are allowed. Code a separate PVSYN= for each synonym.</p> <p>Note: Copybooks resolved using this parameter are not expanded in-line. These copybooks are written to the PDS at the SYSUT3 DD statement for input to the compile step in the SYSLIB concatenation.</p> |
| SITE=FIDE | Optional | <p>Enables wild card substitution in FIS Systematics EXEC API PROC member name before the member is expanded.</p> |

| Parameter | Use | Description |
|-----------|--------------------------|---|
| TYP= | Optional but Recommended | <p>Specifies the library type, data set name, and origin of a like-copy library. Format: TYP=ttt/L/pppppppppp where</p> <p>ttt Library type</p> <p>L Like-copy library data set name</p> <p>pppppppppp Package ID of a participating package (Generated by file tailoring only if cross pollination of copybooks from associated participating packages is in effect.)</p> <ul style="list-style-type: none"> ■ TYP= parameters are created in skeleton file tailoring (CMN\$\$WRT) to identify which application and library type supplied each copybook that is processed. This information is stored in the package master and is used by package audit to discover SYNCH15 errors. ■ If you omit TYP= parameters, CMNWRITE will not fail, but audit will not detect SYNCH15 errors. ■ TYP= parameters have no effect on the library search order of CMNWRITE. <p>Note: If you customize the SYSLIB concatenation in CMN\$\$SYC to include like-copy baseline libraries from other applications, use the following TYP= format to avoid SYNCH15: TYP=ttt/L/aaaa where</p> <p>ttt Library type</p> <p>L Like-copy library data set name</p> <p>aaaa Application ID of the specified library</p> |

Return Codes and Error Messages

| Return Code | Description |
|-------------|---|
| 0 | Successful execution. |
| 4 | Refer to the short and long messages displayed within the job. |
| 6 | Unable to connect to ChangeMan ZMF instance; resubmit job under an active ChangeMan ZMF instance. |
| 8 | Package master I/O error; check all messages displayed within job. |
| 12 | System error; see messages. |



CAUTION! A non-zero return code from CMNWRITE can lead to invalid SYNCH5, SYNCH15, and SYNCH16 audit errors. See ["CMNWRITE and Audit" on page 143](#).

Reporting

The SYSPRINT DD statement for CMNWRITE displays the following information:

- Program version.

- PARM input.
- Keyword parameters input to SYSIN.
- Copybook library search order.
- Summary report of copybook members and hierarchy.

The summary report from the sample CMNWRITE JCL above might look like this:

```
*****
* DDNAME: WRITE.SYSPRINT
*****
ChangeMan(R) ZMF      CMNWRITE - 8.1.4  SUNDAY NOVEMBER 26, 2017  21:51:24
PARM='SUBSYS=6,USER=USER015,'
PARM interpretation: ChangeMan ZMF subsystem "6"
Expansion of uncovered COPY/Include variations

SYSIN: TYP=CPY/CMNTP.S6.ACTP.STG6.#000081.CPY
SYSIN: TYP=CPY/CMNTP.S6.V810.BASE.ACTP.CPY
SYSIN: TYP=CP2/CMNTP.S6.V810.BASE.ACTP.CP2
SYSIN: CMP=CTST.SRC
SYSIN: LNG=COBOL2
SYSIN: PKG=ACTP000081
CMN7500I - Attempting to initiate dialog with started task.
CMN1400I - Session established with ChangeMan ZMF started task.
Library search order: PDS/CMNTP.S6.ACTP.STG6.#000081.CPY          (CPY)
                    PDS/CMNTP.S6.V810.BASE.ACTP.CPY          (CPY)
                    PDS/CMNTP.S6.V810.BASE.ACTP.CP2          (CP2)
LV -MEMBER--- VV.MM --CREATE-- --LAST CHANGED-- SIZE  INIT -USERID- -- L I B R A R Y   N A M E ----- ---FINGERPRINT---
                    --- C O M P O N E N T   D E S C R I P T I O N --- (IF ANY)-----
00 CTST          01.02 2012/07/01 2017/11/26 21:51      7      7 USER015  SYS17330.T215123.RA000.USER0157.SOURCE.H09  AF94653D-000000CD
01 COPYHELO      01.03 2012/07/01 2017/11/26 21:50      1      1 USER015  CMNTP.S6.ACTP.STG6.#000081.CPY  7A3EDE59-00000023
CMN1410I - Session terminated with ChangeMan ZMF started task.
CMN5400I - Time of day at end of job: 21:51:24 - Condition Code on exit: 0
Record count in ==>> 00007
Record count out ==>> 00014
```

In this report, the hierarchy of nested components is indicated by the LV column. If a copybook is not expanded inline into the source and is written to SYSUT3 instead, the LV number is preceded by "N," and message CMN5420I is printed beneath the component list.

Notes

COPY and INCLUDE Variations

This table displays examples of source code COPY and INCLUDE statements that CMNWRITE can detect. This is not an exhaustive list of statements that CMNWRITE can process.

| Format | Action | Comment |
|--------------------------|------------------------------------|------------------|
| 01 COPY ABC. | Expanded | COBOL variation |
| COPY "ABC". | Expanded | COBOL variation |
| COPY ABC | Expanded | Assembler format |
| COPY ABC | Noted as resolved but not expanded | COBOL format |
| COPY ABC REPLACING... | Noted as resolved but not expanded | COBOL variation |
| 01 FIELD-NAME. COPY ABC. | Noted as resolved but not expanded | COBOL variation |
| #INCLUDE <abc> | Expanded | "C" variation |
| #INCLUDE "abc" | Expanded | "C" variation |

CMNWRITE writes copybooks that can be detected and resolved but not expanded to DD statement SYSUT3. Skeleton CMN\$\$WRT builds the SYSUT3 library to temporary data set &&CPYLIBA.

Skeleton CMN\$\$WRT also sets variable CPYLIBA to YES. Skeleton CMN\$\$SYC adds the &&CPYLIBA data set to the SYSLIB concatenation if variable CPYLIBA is YES. In skeletons delivered with ChangeMan ZMF, variable CPYLIBA is not reset to NO until the end of procedures where CMN\$\$SYC might be imbedded multiple times.

Do not customize skeletons to reset variable CPYLIBA to NO until after all imbeds of CMN\$\$SYC that must include &&CPYLIBA in the SYSLIB concatenation.

Recursive Nesting and C++ Headers

Programming language C++ implicitly allows recursive copy structures by requiring each programmed function to include all headers it will use. Compiler directives #ifndef and #define prevent looping in the resulting definitions. It is beyond the scope of CMNWRITE to interpret these compiler directives, so the NOEXPAND program parameter or the EXPAND=NONE SYSIN parameter should be used with C++ source to allow the C++ compiler to resolve these potentially recursive structures.

CMNWRITE must still analyze source and copybooks to provide source/copy information for audit, so beginning with ChangeMan ZMF 5.2, it records the names of all copybooks it encounters in a nest structure. As it begins a new level of a nest, it checks the new copybook name against those that have previously been encountered in this chain. If the name is found, then CMNWRITE assumes that a recursion has been discovered, and it will not search this copybook for copy or include commands.

Modifying Copybook Records With CMNEX016

CMNWRITE processes source code containing CA Librarian -INC statements where the source and copybooks reside in PDS(E) libraries rather than CA Librarian libraries. Exit program CMNEX016 mimics a CA Librarian exit that modifies copybook records included in source with the -INC command.

If CMNEX016 is enabled, it is called by CMNWRITE before each copybook record is written to the file at DD statement SYSOFILE. You can add logic to CMNEX016 to modify or skip copybook records before they are written to SYSOFILE.

The source for CMNEX016 is delivered in the CMNZMF ASMSRC library. See program comments for more information.

SERCOPY - Copy Utility

Program SERCOPY provides enhanced copy services for ChangeMan ZMF batch jobs and internal processes. Some copy functions are provided by proprietary programs, and other functions may be provided by calls to standard IBM copy utilities.

SERCOPY performs these functions:

- Copy partitioned data set member.
- Copy sequential data set.
- Compress Expand compressed listings.

- Add, reset, or update ISPF statistics
- Dynamically reallocate PDS libraries during a copy function.
- SERENQ, Enqueue/dequeue data set

SERCOPY Input

- Partitioned data set or sequential data set.
- Keyword parameters in the PARM statement.
- Member names in the SYSIN statement.

Output

- Members in partitioned data set.
- SYSPRINT output.

Sample JCL

The following is a sample job fragment showing a SERCOPY step.

```
//SERCOPY EXEC PGM=SERCOPY,    *** COPY CTST FROM STAGING
//          REGION=3M,
//          PARM=(' INDSN(CMNTP.S6.ACTP.STG6.#000081.SRC) ',
//          'MEMBER=CTST')
//SYSPRINT DD DISP=(,PASS),DSN=&&LIST00,
//          UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE)
//ABNLIGNR DD DUMMY
//SYSUT2 DD DISP=(,PASS),DSN=&&SOURCE(CTST),
//          UNIT=SYSDA,SPACE=(CYL,(1,2,1),RLSE),
//          DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=0)
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
```

DD Statements

This table describes DD statements for SERCOPY.

| DDNAME | I/O | Purpose |
|----------|--------|---|
| SYSIN | Input | Member list for copies if MEMBER keyword parameter is not coded in the PARM statement |
| SYSPRINT | Output | Report file that displays information from the execution of SERCOPY |
| SYSUT1 | Input | PDS if INDSN and INFILE keyword parameters are not coded in the PARM statement |
| SYSUT2 | Output | PDS if OUTDSN and OUTFILE keyword parameters are not coded in the PARM statement |
| SYSUT3 | I/O | Work data set |
| SYSUT4 | I/O | Work data set |

The data set in DD statement SYSUT1 is dynamically allocated if keyword parameter INDSN or INFILE is coded in the PARM statement.

The data set in DD statement SYSUT2 is dynamically allocated if keyword parameter OUTDSN or OUTFILE is coded in the PARM statement.

PARM Options

The PARM parameter is required in the EXEC statement for SERCOPY. This table describes SERCOPY options that are input through the PARM parameter. **Note** that the default setting for the STATSA/STATSB/STATSE parameters (if these are not specified) is that the copied member will have the same type of statistics as the input member.

| Parameter | Description |
|-----------------|--|
| ABEND | ABEND if error is encountered. |
| ALIAS | Copy all alias entries for members selected. |
| BSAM | Perform internal copy rather than IEBCOPY. |
| COMPRESS(n) | Compress data using compression type n. For low compression n=2. For high compression n=7. |
| CSTATS | Copy existing ISPF statistics and add statistics to copied members if none exist. |
| EXPAND | Decompress data if compressed. |
| FULL | Copy all members. |
| INDSN(dsname) | Specifies DSN for input data set. If this parameter is specified, the input data set name is dynamically allocated. Mutually exclusive with parameter INFILE. |
| INFILE(ddname) | Specifies ddname for input data set. Default is SYSUT1. |
| LIST | List member names in IEBCOPY message output. |
| LMOD | Copy using IEBCOPY COPYMOD. |
| MEMBER(mem,...) | Specifies list of member names to be copied. If this parameter is omitted, the member names are read from SYSIN statements containing one or more member names per line in free form format. If this parameter is omitted and SYSIN is missing or empty, or if MEMBER() is specified, a FULL copy is performed. Members can be renamed during the copy operation by specifying each member name to be renamed in the following format: oldname/newname |
| MFS | Input is MFS data set with non-standard member names. This parameter forces the BSAM option. |
| NOREPL | Do not replace like named members. |
| OSTATS | Reset these ISPF statistics on copied members and add statistics if none exist: Changed MM VV |
| OUTDSN(dsname) | Specifies DSN for output data set. If this parameter is specified, the output data set is dynamically allocated. Mutually exclusive with parameter OUTFILE. |

| Parameter | Description |
|-----------------|--|
| OUTFILE(ddname) | Specifies ddname for output data set. Default is SYSUT2. |
| PDSCOMP | Compress after copy operation. |
| PRINT(ddname) | Specifies ddname for output print data set. Default is SYSPRINT. |
| REALLOC | Reallocate if more space required. |
| RETRY | Retry on x37 abend (compress). This parameter is incompatible with the AUTOOMPX=YES of PDSFAST®. See "PDSFAST" on page 149 |
| RSTATS | Initialize all ISPF statistics on all copied members, including members that did not have statistics. |
| STATSA | Extended ISPF stats will be turned on automatically when the line count reaches 64k. |
| STATSB | Basic ISPF stats will be set. |
| STATSE | Extended ISPF stats will be set. |
| USERID(tsoid) | Specifies the ID for ISPF statistics for copied members. Statistics added to members with CSTATS parameter. Statistics in all members copied with OSTATS parameter. Statistics in all members copied with RSTATS parameter. Statistics added to members with VSTATS parameter. |
| USTATS | Update these ISPF these statistics on copied members and add statistics if none exist: Changed Increment VV by +1 Reset MM to 00 |
| VSTATS | Update these ISPF these statistics on copied members and add statistics if none exist: Increment VV by +1 Reset MM to 00 |

SYSIN Parameters

If the MEMBER keyword parameter is omitted from the PARM statement, the members copied by SERCOPY must be specified in SYSIN records.

- Member names in SYSIN records are coded in free-form format with members listed in the same record separated by spaces or comma.
- Members may be renamed in the copy process by coding the old name and new name separated by a forward slash:

OLDNAME/NEWNAME

- If the MEMBER keyword parameter is omitted from the PARM statement, and no members are specified in SYSIN records, all members in the input PDS are copied to the output PDS.

Return Codes and Error Messages

This table describes return codes for SERCOPY.

| Return Code | Description |
|-------------|---|
| 00-16 | Same as IEBCOPY/IEBGENER |
| 20 | IEBCOPY/IEBGENER ABEND (completion code in R0) |
| 24 | Dynamic allocation error (SVC 99 error in R0) or open error on control statement data set |
| 28 | Input parameter syntax error |

Comments

Automatic Library Reallocation

If the `RETRY` parameter is specified and SERCOPY encounters a space allocation problem while performing a PDS copy function (an x37 condition), it will attempt to recover by compressing the library using IEBCOPY.

If the problem persists and the `REALLOC` parameter is specified, SERCOPY will dynamically reallocate the target library to increase space and/or directory entries.

SERCOPY calls program SERREAL to reallocate the target data set. SERREAL determines the current size of the data set and sets new space allocations based on the following:

- If directory blocks are insufficient, they are increased 50%, plus 8 blocks.
- If directory blocks are *not* the only problem, they are increased 25%, plus 8 blocks.
- If directory blocks are sufficient, then library space is increased on a sliding scale depending on the size of the current data set:
 - If the existing data set is small (10 tracks), space is at least doubled.
 - If the existing data set is large (400 tracks), space is increased by only 20%.

The default unit for the space allocation is blocks (BLKS).

Exit program SEREX001 can be used to override the default generic device (SYSDA) or space unit (BLKS) used by SERREAL in dynamic space reallocation.

Step and JOB Enqueue

ChangeMan ZMF batch jobs file tailored from skeletons will execute one at a time if they target the same output data set with SERCOPY. You can increase the efficiency of ChangeMan ZMF batch processing by enqueueing SERCOPY output data sets at the step level.

To protect the directory of PDS libraries targeted by SERCOPY, skeleton CMN\$\$ENQ is imbedded in each job step that executes SERCOPY. CMN\$\$ENQ catalogs a work data set with `DISP=(MOD,DELETE)` in DD statement SYSUT3. CMN\$\$ENQ uses a consistent rule to create the work data set name from the SERCOPY output data set name.

Only one job is allowed to allocate the same cataloged data set name. If more than one job targets the same library with SERCOPY, only one job will be allowed to allocate the

same work data set name at a time, and all of the other jobs will be made to wait. This effectively single threads all jobs that target the same PDS with SERCOPY.

You may enqueue the SERCOPY output library at the step level by specifying the data set name in the SERCOPY PARM statement:

- 1 Code the SERCOPY output DSN in the PARM statement in subparameter OUTDSN:

```
//COPY1 EXEC PGM=SERCOPY,REGION=3072K,
// PARM='&OPT,OUTDSN(dsname)'
```

The data set name specified in OUTDSN must already be catalogued.

- 2 Delete the SYSUT2 DD statement from the SERCOPY step JCL.
- 3 Remove the imbed for CMN\$\$ENQ from the skeleton that executes SERCOPY.
- 4 Add these two ddnames to the SERCOPY step JCL:

```
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
```

PDSFAST

If you are using PDSFAST with AUTOCMPX=YES, you must remove the RETRY parameter for SERCOPY in the skeleton JCL. You can only use one or the other compression technique (that of PDSFAST or SERCOPY).

Reporting

The SYSPRINT DD statement for SERCOPY displays output from the copy utility used to copy members from the input PDS to the output PDS.

SERPRINT - SYSOUT Compression Facility

ChangeMan ZMF has a facility that makes it possible to eliminate the paper printing and subsequent storage, and access and traceability problems of any application's SYSOUT listings. This facility gathers files destined to SYSOUT queues, concatenates them, and compresses them into a single component of a PDS. You can use this facility regardless of whether or not your shop has converted to ChangeMan ZMF.

Each listing data set normally queued to a SYSOUT class is redirected to a temporary file with accurate DCB attributes and adequate disk space to hold it. The recommended DSNAMES of each passed file is &&LISTxxx, where LIST could be some other recognizable character string but LIST is very meaningful. For example:

```
//SYSPRINT DD SYSOUT=*
```

could be converted to:

```
//SYSPRINT DD DISP=(,PASS),DSN=&&LIST010,
// UNIT=SYSDA,SPACE=(CYL,(1,2),RLSE),
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=2420)
//* or DCB=(RECFM=VBM,BLKSIZE=4088)
```

There could be multiple converted SYSOUT definitions per job step. The only restriction is that each temporary DSNAMES must start with the same character string (LIST) and

suffixed with an alphanumeric string that makes it unique within the job. It is further recommended that the suffix be at least two numeric digits in ascending order with adequate spacing to keep it manageable. The DCB attributes of each passed file are independent of one another. Since SERPRINT is unable to handle VIO data sets, it is required that the files are allocated on non-VIO devices. To accommodate this requirement, a new variable, DEFAULT NON-VIO UNIT NAME has been added to Global Admin panel CMNGGP01. Make sure your SYSPROG/STORAGE ADMIN has designated this UNIT for NON-VIO usage.

The gathering and concatenation of each file is done by component SERPRINT. For example:

```
//SERPRINT EXEC PGM=SERPRINT,COND=EVEN,
//          PARM=(' INDSN(LIST*) ',
//          'OUTFILE(PRINT1,PRINT2)')
//PRINT1 DD DISP(,PASS),DSN=&&LIST,
//          UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
//          DCB=(RECFM=VBM,LRECL=133,BLKSIZE=23476)
//PRINT2 DD SYSOUT=*,DCB=(RECFM=VBM,LRECL=133,BLKSIZE=23476)
```

The specification of COND=EVEN is important. Before, when spooling directly to a SYSOUT queue, you could see all that had been spooled up to the point of any surprising ABEND. With this facility, those passed file contents can only be viewed by SERPRINT's post-processing capabilities and any previous step that ABENDs would normally flush this step unless COND=EVEN has been specified.

The PARM is **critical**. It dictates the input file structure to search for and the output file or files to put the concatenation out to. There are two forms for input file structure:

```
//          PARM=(' INDSN(LIST*) ',...
```

and

```
//          PARM=(' INFILE(STEP10.SYSPRINT),...
```

The use of INFILE is *not recommended* as it refers to procedure step names and DDNAMEs which is rather inflexible. The recommended approach is to use INDSN which dictates the low order node prefix structure to search for. The searching process involves the Scheduler Work Area (SWA) Manager above or below the XA line. Each discovered file is read and each record is converted to VBM - variable blocked machine control characters with trailing blanks truncated.

Each output file DDNAME is specified by the keyword OUTFILE in the PARM. In the recommended example above, DDNAMEs PRINT1 and PRINT2 are targets for the concatenated output. PRINT1 is passed as a sequential file to be compressed. PRINT2 is written directly to a SYSOUT queue for browsing and potential redirection to paper listing or deletion. Each concatenated file is preceded by a banner detailing where it came from. For example:

```
DDNAME: procstep.stepname.ddname
```

As suggested above, *procstep* is the step name (if any) that invoked the procedure; *stepname* is the actual step name within the JCL stream that invoked a program that wrote to DDNAME *ddname*. A trailing statistics record is written detailing records and bytes used. For example:

```
<STATS: RECS=nnnn BYTES=nnnnnnn>
```

The output file that is passed (in this case, PRINT1) is targeted for input to component SERCOPY. The example that follows is in Skeleton notation but could be modified slightly to use symbolic substitution in a cataloged procedure:

```
//COMP      EXEC PGM=SERCOPY,REGION=4M,
//  PARM=' COMP,IN(SYSUT1),OUT(SYSUT2),MEMBER(&CMPNAME),REALLOC'
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=(OLD,DELETE),DSN=####LIST
//SYSUT2   DD  DISP=SHR,DSN=&STGLIB.LST
```

The PARM dictates that a compress (COMP) is to be performed using file SYSUT1 as input, file SYSUT2 as output, and whatever component name substitution for &CMPNAME is file tailored. If the component already exists in the output library, it is replaced by the new one coming in. If an x37 ABEND occurs, it is intercepted internally, an IEBCOPY compress is invoked on the output library, and the compression is re-initiated. If another x37 ABEND occurs, the output library is dynamically reallocated with larger dimensions elsewhere and the compression re-initiated. Without the REALLOC keyword, no reallocation is attempted.

Beginning with release 4.1.0, the calculation method of determining the compression ratio, displayed at the end of a decompressed listing, has been changed.

The formula for calculation of compression percentage follows:

$$C = (T1 - T2) / T1 * 100$$

where:

C = compression percentage

T1 = bytes full

T2 = bytes compressed

If T1 is 1,000,000 bytes and T2 is 600,000 bytes, we get:

$$(1,000,000 - 600,000) / 1,000,000 = 40\% \text{ compression.}$$

In the previous compression routine, using the Huffman algorithm, number of bytes full (T1) was passed using the largest possible size of each record read BEFORE truncating to ragged right.

T1 is used as the summation of bytes into SERCOPY which already has had considerable white space stripped off - often 30%.

The only real statistic that counts is how much space is saved on the disk pack.



NOTE Alg=2 garners compression is nearly as good as the old Huffman algorithm but without the overhead of having to pass the entire file twice. The full expansion to VBM means to ragged right. Formerly, the compression statistics appended to the listing expansion were based on the full number of bytes of the flat right listing concatenation. Now compression is more accurate in that it is based on the actual number of ragged right byte summation.

Browsing Compressed Listings

The ability to browse a compressed listing is implicit within ChangeMan ZMF but can also be used outside by either a specialized CLIST or submitted batch job to decompress the component and subsequently browsing the sequential data set. We deliver a CLIST called CMNBRWCL and associated panels to perform just such a function. It is the user's responsibility to incorporate this ability into their ISPF panel mechanism. You can however simply (from within a ChangeMan ZMF session) use the command TSO %CMNBRWCL and

the selection panel CMNLSTB0 will display. Supply the Dataset and Member and you can see the decompressed listing:

```

CMNLSTB0                      Browse Compressed Listing
Command ==>> _____

Compressed listing library:

Dataset . . . . CMNTP.S6.COMM.STG6.#000001.LST
Member  . . . . COMSRS00      (blank for member selection list)

```

Also note that at the bottom of the decompressed file are statistics within banners detailing the amount of compression attained. The calculation is based on the actual number of bytes needed to hold the compressed file divided by the number of bytes in the composite decompressed file(s). For example:

```

*****
* Compression Statistics          *
*                               *
* Number of records:    494      *
* Bytes expanded:      36,552    *
* Bytes compressed:    8,240     *
* Compression ratio:   77.46%    *
*****

```

There are other vendor products for storing listings on line but they do not discern which is the latest or the one that corresponds to what you have executed in production. Any kind of file that can be sent to a SYSOUT spooling can be incorporated with this mechanism.

Chapter 6

Reports

REXX programs that call XML services generate the reports that are supplied with ChangeMan ZMF. This chapter provides guidelines for the experienced REXX programmer who needs to develop new reports or customize the reports that are shipped with ChangeMan ZMF.

| | |
|---|-----|
| Overview of Online Report Generation | 154 |
| Submitting a Batch Job to Generate a Report | 155 |
| Analysis of a Sample REXX Reporting Program | 155 |
| XML Services Called in Reporting Programs | 163 |

Overview of Online Report Generation

Global administrators, application administrators, and users can run the reports that ship with ChangeMan ZMF:

- The global administrator determines the reports that an application administrator can run. The *ChangeMan ZMF Administrator's Guide* describes the reports that global and application administrators can run and shows how to access them online.
- An application administrator determines the reports that the user can run. The *ChangeMan ZMF User's Guide* describes the reports that users can run and shows how to access them online.

You can request a report online or submit the request as a batch job. For security purposes, both methods require that the TSO userid of the requesting user be passed to the target XML service. Thus, report access is determined by the security authorization of the user who submits the report request.

The steps that the global administrator, application administrator, or user takes to generate ChangeMan ZMF reports online are:

- 1 The administrator or user specifies the appropriate report option and selects the desired report from an online menu:
 - The global administrator selects the target report from the Report Selection List (CMNREPT6) panel (option =A.G.R.2).
 - The application administrator selects the target report from the Report Selection List (CMNREPT6) panel (option =A.A.R.2).
 - The ChangeMan ZMF user selects the target report from the Report Selection List (CMNREPT6) panel (option = 6.1).
- 2 The selected report corresponds to a member of the REXX program library. For example, report 010 corresponds to member CMN010 in the REXX program library. When the ChangeMan ZMF administrator or user selects a report to generate online, ChangeMan ZMF submits a batch job, passing it the name of the target reporting program and the reporting option (application name or pattern, package name or pattern, and so on) that the administrator or user has specified.
- 3 The target REXX program:
 - Validates the input.
 - Sets up the appropriate REXX stem variables.
 - Calls the appropriate XML service, passing it the stem variable to process. The program may call more than one XML service. The XML services that the target reporting program calls are identified in the program comments.
 - Formats the information returned by the target XML service and sends report output to DDname SYSTSPRT. All reports include print-control characters in column 1. Therefore, the SYSTSPRT statement in the CMN\$\$RPT member of the vendor-supplied CMNZMF.SKELS library and in the REPORTS member of the CMNZMF.CNTL library specify the attributes RECFM=FBA and LRECL=133.
- 4 You use a facility such as the Spool Display and Search Facility (SDSF) to view the report.

Submitting a Batch Job to Generate a Report

As an alternative to selecting a report to generate online through the ChangeMan ZMF client, you can submit your own batch job outside of ChangeMan ZMF to generate a report. To do so:

- 1 Customize the REPORTS member of the CNTL library as follows (according to the comments in member REPORTS):
 - Supply a valid JOB statement.
 - Supply the appropriate variables for the target report.
 - Specify the appropriate library names in the DD statements.
- 2 Submit the job.



NOTE The ChangeMan ZMF ISPF client does not have to be running, but the started task does. The batch job connects directly to the SERNET started task.

Notes on the Batch JCL

Note the following items about the JCL that is submitted for execution when you select a report to generate online in ChangeMan ZMF or submit a batch job:

- The SYSEXEC DD concatenation identifies where the REXX reporting programs are located. Not all installations have the optional REXX compiler. Therefore, the REXX source programs for the batch reports are shipped. If you have the REXX compiler at your installation, you can compile these source programs and store them in the CEXEC library if you wish. If you do, be sure to add the name of the CEXEC compiler library to the top of the SYSEXEC DD concatenation.
- You always need a SER#PARM DD statement to identify the system to which you want to connect.
- TCP/IP is used for communication across address spaces when a report is run. TCP/IP messages are written to SYSPRINT.
- Diagnostic messages are written to SERPRINT.
- Report output is written to SYSTSPRT.
- If an abend should occur when you are running a report, information about the abend is written to SERABEND and SYSABEND.

Analysis of a Sample REXX Reporting Program

All of the REXX reporting programs that are shipped with ChangeMan ZMF have the same structure. We describe program CMN010 in this section to illustrate how the reporting programs work and give you guidelines for customizing them if you wish. You can also use the program as a model to develop your own reporting programs.

The source code for report 010, Summary of Planned and Unplanned Packages, is in member CMN010 of the REXX program library.

Introductory Comments

The following code excerpt is typical of the introductory commentary in each reporting program. The introductory comments:

- 1** Have the word REXX on the first line (required in all REXX programs).
- 2** Identify the report number and title. In this example the report number is *CMN010* and the title is *Summary of Planned and Unplanned Packages*.
- 3** Identify the XML services that are called to provide information for the report. Two XML services are called in this example: PARMs and PACKAGE. See the *XML Services User Guide* for a description of the XML services that the reporting programs call.

- 4 Identify the parameters that the program passes to the SERXMLRC program, which handles communication between the reporting program and the target XML service.

```

/* REXX                                                                 */ 1
/* *****                                                             */
/* Copyright 2003-2012 (C) SERENA Software, Inc.                       */
/* Licensed material. All rights reserved.                             */
/* ChangeMan is a registered trademark of SERENA (R) Software Inc.     */
/* *****                                                             */
/* USE OF THE SAMPLE CODE CONTAINED HEREIN IS SUBJECT TO THE TERMS   */
/* CONDITIONS OF THE LICENSE AGREEMENT LOCATED IN THE MEMBER LICENSE */
/* *****                                                             */
/* Date      Author      Reason                                       */
/* 2003-06-01 Serena    Original version                             */
/* *****                                                             */
/* REXX CMN010 Summary of Planned and Unplanned Packages              */ 2
/*                                                                 */
/* This report makes use of two XML Services                           */ 3
/*                                                                 */
/* Service    Scope    Message    Description                         */
/*                                                                 */
/* 1 PARMS     APL      LIST       Obtain the list of Appl. names  */
/* 2 PACKAGE   SUMMARY SERVICE   Obtain counts about Package types */
/*                                                                 */
/* Parameters                                     */ 4
/*                                                                 */
/* Application name      1 to 4 character mnemonic which may       */
/*                                                                 */
/*                                                                 */
/*                                                                 */
/*                                                                 */
/*                                                                 */
/* Subsystem letter     1 character indicative of the ChangeMan    */
/*                                                                 */
/*                                                                 */
/*                                                                 */
/* TSO userid           1 to 8 character TSO id used to perform    */
/*                                                                 */
/*                                                                 */
/* Test switch          An indicator with the value 'T' which      */
/*                                                                 */
/*                                                                 */
/*                                                                 */
/*                                                                 */
/*                                                                 */
/*                                                                 */
/* *****                                                             */

```

Mainline Program Logic

The mainline logic of all the reporting programs is the same. Mainline program logic performs the following functions:

- 1 Gets the input from the user. (See ["Getting User Input" on page 158.](#))

- 2 Calls a subroutine in the program to validate the user input. (See ["Validating User Input" on page 158.](#))
- 3 Calls a subroutine in the program to initialize the variables that will be passed to the target XML service. (See ["Initializing Variables" on page 160.](#))
- 4 Calls a subroutine to set up the XML service call. (See ["Setting Up the XML Service Call" on page 161.](#))
- 5 Calls a subroutine that makes the call to the target XML service. (See ["Calling the Target XML Service" on page 161.](#))
- 6 Calls a subroutine to format and print the output.
- 7 Calls a subroutine to disconnect from ChangeMan ZMF.

These subroutines appear in the Subroutines section of the reporting program. Selected subroutines that are called in our sample CMN010 program are described below. You may want to consider the annotations given for the code excerpts below while you are looking at a complete program source listing.

Getting User Input

The ARG instruction is used to get the user input. The variable names in the ARG instruction correspond to XML tags in the XML services that the program will be calling.



CAUTION! Be sure to use variable names in the program that differ from the XML tags to avoid double substitution of variable values.

Here's the ARG instruction used in program CMN010:

```
/* Read input parms */
arg appname subname tsoname tst .
```

The arguments are:

- The name of the target application, a pattern that identifies all applications that match the pattern, or an asterisk that identifies all applications that are defined to the target ChangeMan ZMF subsystem.
- The name of the target ChangeMan ZMF subsystem.
- The ID of the requesting TSO user.
- A test switch (optional). This switch, if present, requests that diagnostic trace information be written to the SERPRINT DDname.

Validating User Input

The program calls the common program CMN000 to validate the user input parameters.

The parameters may vary slightly from report to report but as a general rule they consist of an application name/package number, a subsystem letter, a userid and finally, a test option which should only be used under the direction of Customer Support. For example parts of the validation in CMN000 are as follows:

```

/* Validate ParmS */
Validate_Parms:

  /* applname must be 1 to 4 characters if present */ 1

if length(applname) > 4 then
  do
    call Error_Message 'Application name too long'
  end

...

  /* subsystem must be 1 character from the approved list */ 2
subsyslt = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789#@$ '
if length(subname) > 1 | pos(subname,subsyslt) = 0 then
  do
    call Error_Message 'Subsystem Letter invalid'
  end

...

  /* userid must be present and 1 to 8 chars */ 3
if length(tsoname) > 8 | length(tsoname) < 1 then
  do
    call error_message 'Userid invalid'
  end

...

  /* test switch must be 'T' or 'X' if present */ 4
/* it can also be absent or denoted with placeholder '.' */

Select
  when tst='T' then nop
  when tst='X' then nop
  when tst='.' then nop
  when tst=' ' then nop
  otherwise
  do
    call Error_Message 'Test Switch invalid: must be T, X or blank'
  end
end

```

In this example:

- 1** The application name must be from 1 to 4 characters if it is supplied.
- 2** The ChangeMan ZMF subsystem name must be 1 character or blank.
- 3** The TSO userid of the requester must be present and be from 1 to 8 characters.
- 4** The test switch, if passed, must have the value T or X. If you are having problems, Customer support may ask you to specify the test switch. If so, diagnostic messages are written to the SERPRINT DDname.

Initializing Variables

The following code excerpt shows the subroutine that program CMN010 uses to initialize the variables:

```

/* Initialize Variables */

Init_Variables:

  if appname = '.' | length(appname) = 0 then /* appl name default */ 1
    do
      appname = '*'
    /* end if */

  if subname = '.' | length(subname) = 0 then /* subsys default */ 2
    do
      subname = ' '
    end
  /* end if */

  /* initialize grand totals */ 3

  GTSimple = 0
  GTComplex = 0
  GTSuper = 0
  GTPart = 0
  GTPPerm = 0
  GTPTemp = 0
  GTUPerm = 0
  GTUTemp = 0

  /* set date /time for report header */

  headdate=date()
  headtime=time()

  /* set page counter variables */

  pchar = ' ' /* asa required, make null if not FBA */
  linect = 99 /* expire the line counter to force headings on page 1 */ 4
  lines = 55 /* print 55 data lines per page */
  linelen = 132 /* report line length (lrecl-1) */
  pagect = 0 /* page counter */
return

```

In our sample CMN010 program:

- 1 The user can input a period or asterisk for the application name or leave it blank. The program treats any of these values as if the user had specified the asterisk to signify all applications that are defined in the target ChangeMan ZMF subsystem.
- 2 If the subsystem ID is presented to the program as a period or is not supplied, the program assumes a blank subsystem ID.
- 3 The program initializes the grand totals to be printed in the report.
- 4 The program sets up line and page counters.

Setting Up the XML Service Call

All XML services expect the calling REXX reporting program to pass a stem variable. Here's the subroutine in our sample CMN010 program that sets up the stem variable that is to be passed to the PARMS APPL LIST service. Note that the variable names used in the reporting program should differ from the XML tag names to avoid double-substitution of variable values.

```

/* Set variables for XML call */

Init_XMLStem1:
  rxrc          = 0          /* initialize our return code */
  stem          = "SER1."   /* set outgoing stem name */
  SER2.         = ""        /* initialize outgoing stem */
  SER1.         = ""        /* initialize outgoing stem */
  SER1.Subsys   = subname   /* subsystem name to query */
  SER1.Userid   = tsoname   /* userid */
  SER1.Test     = tst       /* set test value */
  SER1.Product  = "CMN "    /* set product */
  SER1.Service  = "PARMS"   /* set service*/
  SER1.Message  = "LIST"   /* set message */
  SER1.Scope    = "APL"     /* set scope */
  SER1.applname = applname  /* set application name */
                  /* set result set to return */
  SER1.includeInResult.1 = "applName"

Return

```

A userid of the following form is required on any stem that is passed for security checking:
SER1.Userid = *tsoname* /* userid to use for security validation */

You can code the *tsoname* as a string, for example, "abcdefg", or use the REXX function USERID(), which will substitute the current TSO userid or, in the case of a batch job, the TSO userid of the job submitter. SERNET will not allow the *tsoname* to differ from the real userid of the user who is executing or submitting the job.

Calling the Target XML Service

This Serxmlrc subroutine invokes the SERXMLRC program to make the target XML service call. SERXMLRC is the interface between all ChangeMan ZMF reporting programs and the target XML services. SERXMLRC is a member of the SERCOMC.LOAD library. You must use an ADDRESS instruction in REXX to link to this program through the LINKMVS host command environment.

SERXMLRC expects a REXX stem variable as input, and constructs the service request block from the information that is supplied in the stem variable.

After the target service completes execution, the reporting program checks the return code from the service call and processes the results that the service returned if the call was successful.

Note that the SERXMLRC program does not tailor the result set returned by the target XML service. Thus, the statements following the call to SERXMLRC save only the fields that the reporting program needs (the application name or names, in this example) and drops the returned stem variable to minimize the storage used by the REXX variable pool.

Here's an excerpt from the CMN010 sample program. The excerpt shows:

- 1 The instruction that calls the SERXMLRC subroutine in the program.

- 2 The code for the Serxmlrc subroutine, which invokes the SERXMLRC interface program.
- 3 Drop the stem variable to conserve storage.

```

/* make first xml service call */

call Serxmlrc 1

/* for each application returned perform 2nd XML call */

do jx=1 to SER1.result.0
  call Init_XMLStem2          /* set up 2nd XML call */
  call Serxmlrc              /* make 2nd XML call */
  if rxrc=0 then call Output_result /* if ok, print out result */
end

/* Print out totals */

call Output_Totals

/* terminate ZMF session */

call Disconnect

...

Serxmlrc: 2
  address LINKMVS "SERXMLRC stem"
  rxrc=rc
  if rxrc<>0 then call Diagnose_Error
Return

...

/* Disconnect and set return code */

Disconnect:

  arg exitcode
  if exitcode = ' ' then exitcode = '0'
  call Init_XMLstem0
  call Serxmlrc
  drop SER0. 3
  exit exitcode

```

Diagnosing Errors and Formatting Report Output

The error and output-formatting subroutines in the reporting programs are fairly straightforward, so we do not comment on them here:

- Report output is written to the SYSTSPRT DDname.
- Diagnostic messages are written to the SERPRINT DDname.

Use an output display facility such as SDSF to view report output.

Disconnecting from ChangeMan ZMF

To explicitly disconnect from ChangeMan ZMF, you must issue an XML disconnect service call with the following service, scope, and message attributes:

```
<service name="      ">
<message name="DISCONCT">
<scope name="SERVICE">
```

Note that a blank is required in the service name attribute.

See the disconnect code above in the frame. You must ensure that you exit the REXX program if a nonzero return code appears in the reply message. Do not issue a second disconnect request or an infinite loop may result.

The following return codes, which signify REXX environmental errors, can originate from calls to SERXMLRC:

| Return Code | Meaning |
|-------------|---|
| 24 | Error returned from the IRXEXCOM call. |
| 28 | Error in loading IRXEXCOM the first time. |
| 32 | No input stem variable was passed. |

None of these return codes sets a message as there may be no way to pass a message back by means of the stem variable. All other errors should pass through the STATUSxxx stem variable.

XML Services Called in Reporting Programs

Refer to the *ChangeMan ZMF XML Services User's Guide* for a description of the XML services that the ChangeMan ZMF reporting programs call.

Appendix A

Installation Jobs and Transaction Codes

This appendix shows how ChangeMan ZMF installation jobs run in a series that is distributed across D or DP instances and P instances (or against production libraries in a DP instance) to distribute, install baseline, backout, and revert change packages.

| | |
|--|-----|
| X Node Data Sets | 166 |
| Installation Jobs | 167 |
| Other CMNBATCH Transaction Codes | 170 |

X Node Data Sets

When installation JCL is created for a change package, a PDS is created for each installation site. The name of the PDS follows this format:

devmodel.X.sitename

where:

devmodel **DEV MODEL DSNAME** on the **Global Parameters - Part 6 of 7** panel (CMNGGP06)

X Fixed node name to distinguish the library from package staging libraries.

sitename **Site** on the **function:SITE INFORMATION** panel (CMNPRSTI) for the package

Note: There will be no *sitename* node for packages residing on an A instance.

This library is often referred to as the *X node data set*.

The members in this library contain all of the jobs associated with the distribution, installation, baseline ripple, DB2 bind, backout, revert, and temporary package component delete processes for the package. Member names in the X node data set follow this naming convention:

appltpp

where:

appl Three or four character application mnemonic

tt Two character transaction code

pp Last digits of the package number:

- Two digits where the application mnemonic is four characters
- Three digits where the application mnemonic is three characters

- The JCL in each X node data set member is file tailored from the skeleton with name CMNtt, where tt is the two character transaction code. (The name of the skeleton that is actually used may have a suffix, such as *I* if the ChangeMan ZMF IMS Option is licensed.)
- Installation jobs are often referred to by their transaction code. For example, "the 30 job" means the job for transaction code 30 that was file tailored from the CMN30 skeleton and is in X node data set member appl30pp.
- Before customization, installation job names follow the same naming convention as the X node data set members that contain them. Exit program CMNEX008 can be used to customize the job names, but the member names in the X node data set stay the same.

Since a P instance site may not be on the same LPAR or have shared DASD with the D or DP instance where the X node data set is created, a copy of each X node data set is

transmitted to its target site in the distribution phase of the package installation process. The naming convention for the transmitted production X node data set is:

prdmode1.X.sitename

where:

prdmode1 **PRD MODEL DSNAME** on the **Global Parameters - Part 6 of 7** panel (CMNGGP06)

X Fixed node name to distinguish the library from package staging libraries.

sitename **Site** on the **function:SITE INFORMATION** panel (CMNPRSTI) for the package

The contents of the production X node data set are exactly the same as the development X node data set for the same site.

Installation Jobs

The table in this section shows the installation jobs for a package created on a D or DP instance.

- Each job is described by two columns:
 - **Job** - The two character transaction code
 - **Action** - What the job does
- There are two sets of columns in the table:
 - **Development Center** - Jobs that run on the A, D, or DP instance.
 - **Production Site** - Jobs that run on a P instance.

| Development Center | | Production Site | |
|--------------------|---|-----------------|--------|
| Job | Action | Job | Action |
| 10 | <ul style="list-style-type: none"> ■ Package is audited and/or frozen. ■ Jobs are created in ...X.&node. ■ Package is approved. ■ Job 10 is submitted to initiate the distribution. ■ CMNBATCH transaction 10 says distribution initiated and status is changed to DIS. ■ Vehicle is asked to submit job 11 at remote site. | | |

| Development Center | | Production Site | |
|--------------------|--|-----------------|--|
| Job | Action | Job | Action |
| 11 | Staging libraries are sent to remote site. | 10 | <ul style="list-style-type: none"> ■ Staging libraries are received including QSAM package master. ■ Job 11 is submitted. |
| | | 11 | <ul style="list-style-type: none"> ■ CMNBATCH transaction 11 overlays package records (on PM) with QSAM package master; proper node record is time stamped; status is DIS. ■ Job 14 is submitted. (Only if IEBCOPY is not used.) |
| | | 14 | Job 14 requests vehicle to submit 15 at DEV site. |
| | | 17 | Job 17 is submitted if external (not internal) scheduler is used. |
| | | 18 | Job 18 requests vehicle to submit 19 at DEV site. |
| 15 | Job 15 is submitted. (Only if IEBCOPY is used.) | | |
| 15 | CMNBATCH transaction 15 stamps acknowledgment of distribution. | | |
| 19 | Notification to package creator that distribution failed. | | |
| | | 21 | Perform DB2 bind for production installation. (INSTALL IN PROD = YES). |
| | | 20 | Job 20 is submitted to check if package was previously installed, if not, then it begins installation. |
| | | 20 | CMNBATCH transaction 20 changes package status to INS. |
| | | 20 | Job 24 is submitted. (Only if IEBCOPY is not used.) |
| | | 20t | If Temporary, Job 20t runs to install members into Temporary libraries. |
| | | 24 | Requests vehicle to submit 25 at DEV site. |
| | | 28 | Requests vehicle to submit 29 at DEV site. |
| 25 | CMNBATCH transaction 25 changes package status to INS. | | |
| 29 | Notification to package creator that installation failed. | | |
| 25 | If Permanent, Job 30 is submitted. | | |
| | | 30 | Job 30 is submitted if system environment is ALL. |

| Development Center | | Production Site | |
|--------------------|---|-----------------|--|
| Job | Action | Job | Action |
| 30 | CMNBATCH transaction 30 changes package status to BAS and ripples the baseline. | | |
| 30 | Delete members from promotion libraries based on promotion level and library type. | | |
| | | 31 | If Temporary, Job 31 runs to delete members from temporary libraries. |
| | | 31t | CMNBATCH transaction 31 changes package status to TCC (Temporary Change Cycled) and date/time stamp. Submit job 35. |
| | | 32 | Performs DB2 bind for production installation (INSTALL IN PROD = NO). |
| | | 34t | Requests vehicle to submit 35t at DEV site. |
| 35t | Package status updated to TCC and date/time stamped when all remote sites have been cycled. | | |
| | | 38t | Requests vehicle to submit 39t at DEV site. |
| 39t | Notification to package creator that the package cycle failed. | | CASE: A permanent change must be backed out. Operator makes human decision to back out (full) particular package. Enters backout reasons on panel and ChangeMan ZMF instance copies package to same flat file that was sent from development center. Job 50 is submitted. |
| | | 49 | Job 21 runs the DB2 bind for production backout (INSTALL IN PROD = YES). |
| | | 50 | <ul style="list-style-type: none"> ■ Backs out the change by copying back from BKUP Libraries. ■ Changes package status to BAK. ■ Job 54 is submitted if IEBCOPY is used, else job 51. |
| | | 50 | If system environment is ALL, job 55 is submitted. |
| | | 51 | Job 51 transmits a QSAM package master to the development center and requests a vehicle to submit job 54. |

| Development Center | | Production Site | |
|--------------------|--|-----------------|---|
| Job | Action | Job | Action |
| 54 | <ul style="list-style-type: none"> ■ Reads flat package and transmits reasons. ■ Updates backout reasons into correct package. | | |
| 55 | Job 55 is submitted to reverse ripple the baseline if all remote sites are backed out. | | |
| 55 | Status is changed to BAK; * node record is date and time stamped. | | |
| | | 56 | Job 32 runs the DB2 bind for production backout (INSTALL IN PROD = NO). |
| | | 58 | Job 58 requests vehicle to submit 59 at DEV site. |
| 59 | Notification to package creator that package backout failed. | | |
| | | 64 | Job 64 requests vehicle to submit 65 at DEV site. |
| 65 | Status is changed to DEV. | | |

Other CMNBATCH Transaction Codes

The following table shows other CMNBATCH transaction codes that occur on the development center and production site.

| CMNBATCH Transaction | Explanation |
|----------------------|---|
| 05 | Submits a job based on: <ul style="list-style-type: none"> ■ STE=site ■ NOD=node ■ SUB=jobname |
| 65 | Reverts package back to development: <ul style="list-style-type: none"> ■ Reset general component freeze flag ■ Reset all major date/time stamps ■ Set revert date/time stamp at remote site |
| 80 | <ul style="list-style-type: none"> ■ Promotes or demotes a package ■ Checks out components with or without package association |
| 90 | Activates a component |
| 92 | Deletes staging libraries |
| 93 | Resynchronizes the implementation calendar |
| 94 | Deletes change package records |
| 96 | Decrements the Implementation Calendar when packages are deleted |
| 99 | This transaction is invoked to notify you any time there is a job failure. |

Appendix B

Skeleton Hierarchy

This appendix shows how ChangeMan ZMF ISPF skeletons are imbedded in other skeletons. ISPF file tailoring processes this hierarchy of skeletons to build ChangeMan ZMF job JCL.

This chapter is intended to provide information that will help customers modify ChangeMan ZMF base product functions and build custom functions to support change management processes at their company.

| | |
|--|---------------------|
| Introduction | 172 |
| Analyzing Skeleton Imbeds | 172 |
| Standard ChangeMan ZMF Skeletons | 174 |
| IMS Option Skeletons | 199 |

Introduction

The tables in this chapter present skeletons in the following manner:

- Imbedded skeletons are usually listed in the order they occur.
- If a skeleton is imbedded more than once at the same hierarchical level, it is listed only once.
- In some cases where a skeleton is imbedded more than once at the same hierarchical level, the first occurrence is skipped in favor of listing a later imbed that better conveys the functional structure of the hierarchy.
- Most of the subordinate skeletons listed in this chapter are conditionally imbedded. They are processed by file tailoring only if conditions in)SEL statements are satisfied. You must examine the "calling" skeleton and determine the value of variables in)SEL statements to determine if an imbedded skeleton will be processed.
- Some skeletons that contain no imbeds and are not imbedded in other skeletons have been omitted from this chapter.

Analyzing Skeleton Imbeds

The information in this chapter was assembled by analyzing skeleton libraries using the ISPF Search-For Utility (=3.14).

You can analyze skeleton imbeds yourself by searching skeleton libraries for)IM, the operator in the ISPF skeleton imbed statement:

```

ISRSFSPR                               Search-For Utility
Command ===> _____

Search String . . )IM _____

ISPF Library:
  Project . . . _____
  Group . . . _____ . . . _____ . . . _____
  Type . . . _____
  Member . . . _____ (Blank or pattern for member selection list,
                          "*" for all members)

Other Partitioned, Sequential or VSAM Data Set:
  Data Set Name . . . CMNTP.CMN810.SKELS(*) _____
  Volume Serial . . . _____ (If not cataloged)

Listing Data Set . . . USER015.SRCHFOR.LIST _____
Data Set Password . . _____ (If Search-For data set password protected)

Enter "/" to select option                Execution Mode                Output Mode
_ Specify additional search strings        1 1. Foreground                1 1. View
_ Mixed Mode                              2. Batch                        2. Browse
_ Bypass selection list

```

The following panel shows the results of the search:

```

000001 1 ISRSUPC - MVS/PDF FILE/LINE/WORD/BYTE/SFOR COMPARE UTILITY- ISPF
000002 LINE-# SOURCE SECTION SRCH DSN: CMNTP.CMN810.SKELS
000003
000004
000005 CMN$$ACB ----- STRING(S) FOUND -----
000006
000007 1 /**)IM CMN$$ACB
000008 30 /**)IM CMN$$ACB END
000009
000010 CMN$$ARE ----- STRING(S) FOUND -----
000011
000012 1 /**)IM CMN$$ARE
000013 245 )IM CMN$$XVP
000014 247 /**)IM CMN$$ARE END
000015
000016 CMN$$ASM ----- STRING(S) FOUND -----
000017
000018 1 /**)IM CMN$$ASM
000019 25 )IM CMN$$SYC
000020 57 )IM CMN$$OPT
000021 58 /**)IM CMN$$ASM END
000022
000023 CMN$$AUD ----- STRING(S) FOUND -----
000024
000025 1 /**)IM CMN$$AUD
000026 14 )IM CMN$$SPR
000027 114 )IM CMN$$SPR
000028 123 )IM CMN$$XVP
000029
000030 CMN$$BBL ----- STRING(S) FOUND -----
000031
000032 1 /**)IM CMN$$BBL
000033 7 )IM CMN$$CBL
000034 10 )IM CMN$$PBL
000035
000036 CMN$$BDT ----- STRING(S) FOUND -----

```

Notice that nearly all ChangeMan ZMF skeletons begin with a JCL comment containing the skeleton name:

```
/**)IM CMN$$ASM
```

By searching backward through JCL generated by file tailoring from ChangeMan ZMF skeletons, you can get information about the hierarchy of imbeds that was used by ISPF file tailoring to build the JCL. However, there is no way to tell from the JCL comments if a skeleton was imbedded in a previously listed skeleton or if both were imbedded in another skeleton and processed serially.

This JCL fragment from a stage job shows that skeletons CMN\$\$DSN, CMN\$\$JBL, CMNCOB2, CMN\$\$VAR, CMN\$PARM, and CMN\$\$WRT were processed by ISPF file tailoring

to build the JCL. In fact, skeletons CMN\$\$VAR and CMN\$\$XSC are imbedded in procedure skeleton CMNCOB2, and skeleton CMN\$PARM is imbedded in CMN\$\$VAR.

```

000008 /**
000009 /** JOB REQUESTED BY USER015 ON 2015/02/10 AT 15:17
000010 /**
000011 /**)IM CMN$$DSN
000012 /**)IM CMN$$JBL
000013 //JOBLIB DD DISP=SHR,DSN=CMNTP.CMN810.C6.LOAD
000014 // DD DISP=SHR,DSN=CMNTP.SER810.C6.LOAD
000015 // DD DISP=SHR,DSN=CMNTP.CMN810.LOAD
000016 // DD DISP=SHR,DSN=CMNTP.SER810.LOAD
000017 /**)IM CMNCOB2
000018 /**)IM CMN$$VAR
000019 /** USROP01 = Y
000020 /**)IM CMN$PARM
000021 /** SEL = AND OBJLIB = Y
000022 /**)IM CMN$$XSC
000023 //SERCOPY EXEC PGM=SERCOPY, *** COPY ACPSRCEE FROM STAGING
000024 // REGION=3M,
000025 // PARM=( ' INDSN(CMNTP.S6.ACTP.STG6.#000032.SRC) ',
000026 // 'MEMBER=ACPSRCEE' )
000027 //SYSPRINT DD DISP=(,PASS),DSN=&&LIST00,
000028 // UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
000029 // DCB=(RECFM=FBM,LRECL=121,BLKSIZE=0)
000030 //ABNLIGNR DD DUMMY
000031 //SYSUT2 DD DISP=(,PASS),DSN=&&SOURCE(ACPSRCEE),
000032 // UNIT=SYSDA,SPACE=(CYL,(1,2,1),RLSE),
000033 // DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=0)
000034 //SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
000035 //SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
000036 /**)IM CMN$$WRT
000037 //WRITE EXEC PGM=CMNWRITE, *** PARSE/EXPAND COMPONENT ACPSRCEE
000038 // COND=(4,LT),
000039 // PARM=( ' SUBSYS=6,USER=USER015 ',
000040 // '' )
000041 /**)IM CMN$$SPR
000042 //SER#PARM DD DISP=SHR,DSN=CMNTP.SER810.C6.TCPIPORT
000043 //SYSPRINT DD DISP=(,PASS),DSN=&&LIST10W1,
000044 // UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE)
000045 /**)IM CMN$$SYC
000046 //SYSLIB DD DISP=SHR,DSN=CMNTP.S6.ACTP.STG6.#000032.CPY
000047 // DD DISP=SHR,DSN=CMNTP.S6.V810.BASE.ACTP.CPY

```

Standard ChangeMan ZMF Skeletons

The skeletons in this section are used to build JCL for batch jobs submitted by the base ChangeMan ZMF product.

Checkout Skeletons

These skeletons are used to build JCL for checkout jobs.

| Skeleton | Purpose |
|------------|--|
| CMN\$\$CKO | Check out components from baseline/promotion libraries |
| CMN\$\$SSV | Save prior staging version of target components |

| Skeleton | Purpose |
|-----------------|--|
| CMNSRVC | Invoke batch services |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$CKA | Check out components to a personal library |
| CMN\$\$PAN | Check out components to CA Panvalet development libraries |
| CMN\$\$LB2 | Check out components to CA Librarian development library |
| CMN\$\$LIB | Check out components from CA Librarian baseline libraries |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$XVP | Send notifications to an IP address and port #. |

Stage Skeletons

These skeletons are used to build JCL for stage jobs.

| Skeleton | Purpose |
|-----------------|---|
| CMNASM | Procedure to process assembler source code |
| CMN\$\$VAR | Process user options and/or initialize other variables |
| CMN\$PARAM | Set global (i.e. shop standard) compile, link, etc. parameters. |
| VAR\$xxxx OPT | Optional include where xxxx is application mnemonic for general application specific staging variables. |
| PRM\$xxxx OPT | Optional include where xxxx is application mnemonic to set application-specific compile, link, etc. parameters. |
| CMN\$\$EAR | Ensure nothing has changed since autoresolve job was generated. |
| CMN\$\$XSC | Extract source code member from baseline library during recompile or from staging library during staging. |
| CMN\$\$WRT | Parse source code and expand copy structures, including COPY, EXEC SQL, ++INCLUDE and -INC. |
| CMN\$\$SPR | Include SER#PARAM DD statement |

| Skeleton | Purpose |
|-----------------|--|
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$PDB | Perform a DB2 precompile |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$CIC | Perform a CICS precompile |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$ASM | Translate ASSEMBLER source code |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$OPT | Build in-stream compiler options into ddname(s) |
| CMN\$\$OBJ | Process object code |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$SSI | Process link-edit control cards |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$LNK | Link edit a program |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$OBL | Process object code |
| CMN\$\$SYL | Build the SYSLIB link-edit concatenation |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN\$\$ILL | Build the concatenation used during relink |
| CMN\$\$ILC | Build the sysin concatenation for ILIC processing |
| CMN\$\$PAS | Synch package with area for generate and auto-resolve |
| CMN\$\$DBC | Copy DB2 DBRM to a DBR staging library |
| CMN\$\$CND | Check for a condition code failure |
| CMN90 | Activate components staged in batch |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$RMS | Reset PDF statistics in a partitioned dataset |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN\$\$TRK | Track the success/failure of the generate components |

| Skeleton | Purpose |
|-----------------|---|
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNCOBOL | Procedure to process OS/VSE COBOL source code |
| CMN\$\$VAR | Process user options and/or initialize other variables |
| CMN\$PARM | Set global (i.e. shop standard) compile, link, etc. parameters. |
| VAR\$xxxx OPT | Optional include where xxxx is application mnemonic for general application specific staging variables. |
| PRM\$xxxx OPT | Optional include where xxxx is application mnemonic to set application-specific compile, link, etc. parameters. |
| CMN\$\$EAR | Ensure nothing has changed since autoretrieve job was generated. |
| CMNTLNGN | Generate a TELON COBOL, COBOL2 or PLI program for Compilation in a ChangeMan ZMF job stream. (TELON 2.0c) |
| CMNTLNPI | Print a panel image from PI data in the TELON TDF |
| CMNTLNRS | Resolve PDS COPY statements and CA Panvalet / CA Librarian Includes via TELON |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$XSC | Extract source code member from baseline library during recompile or from staging library during staging. |
| CMN\$\$WRT | Parse source code and expand copy structures, including COPY, EXEC SQL, ++INCLUDE and -INC. |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$PDB | Perform a DB2 precompile |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnid |
| CMN\$\$CIC | Perform a CICS precompile |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$COB | Compile COBOL source code |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$OPT | Build in-stream compiler options into ddname(s) |
| COM\$\$OBJ | Process object code |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$SSI | Process link-edit control cards |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$LNK | Link edit a program |

| Skeleton | Purpose |
|-----------------|--|
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$OBL | Process object code |
| CMN\$\$SYL | Build the SYSLIB link-edit concatenation |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnlid |
| CMN\$\$ILL | Build the concatenation used during relink |
| CMN\$\$ILC | Build the sysin concatenation for ILIC processing |
| CMN\$\$PAS | Synch package with area for generate and auto-resolve |
| CMN\$\$DBC | Copy DB2 DBRM to a DBR staging library |
| CMN\$\$CND | Check for a condition code failure |
| CMN90 | Activate components staged in batch |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$RMS | Reset PDF statistics in a partitioned dataset |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN\$\$TRK | Track the success/failure of jobs submitted by generate |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNCOB2 | Procedure to process COBOL2 source code |
| CMN\$\$VAR | Process user options and/or initialize other variables |
| CMN\$PARAM | Set global (i.e. shop standard) compile, link, etc. parameters. |
| VAR\$xxxx OPT | Optional include where xxxx is application mnemonic for general application specific staging variables. |
| PRM\$xxxx OPT | Optional include where xxxx is application mnemonic to set application-specific compile, link, etc. parameters. |
| CMN\$\$EAR | Ensure nothing has changed since autoresolve job was generated. |
| CMNTLNGN | Generate a TELON COBOL, COBOL2 or PLI program for Compilation in a ChangeMan ZMF job stream. (TELON 2.0c) |
| CMNTLNPI | Print a panel image from PI data in the TELON TDF |

| Skeleton | Purpose |
|-----------------|--|
| CMNTLNRS | Resolve PDS copy statements and CA Panvalet / CA Librarian includes via TELON |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$WRT | Parse source code and expand copy structures, including COPY, EXEC SQL, ++INCLUDE and -INC. |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$PDB | Perform a DB2 precompile |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN\$\$CIC | Perform a CICS precompile |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$CO2 | Compile COBOL2 source code |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$OPT | Build in-stream compiler options into ddname(s) |
| COM\$\$OBJ | Process object code |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$SSI | Process link-edit control cards |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$LNK | Link edit a program |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$OBL | Process object code |
| CMN\$\$SYL | Build the SYSLIB link-edit concatenation |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN\$\$ILL | Build the concatenation used during relink |
| CMN\$\$DBC | Copy DB2 DBRM to a DBR staging library |
| CMN\$\$CND | Check for a condition code failure |
| CMN90 | Activate components staged in batch |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$RMS | Reset PDF statistics in a partitioned dataset |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |

| Skeleton | Purpose |
|-----------------|--|
| CMN\$\$TRK | Track the success/failure of jobs submitted by generate |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNMAPGN | Procedure to process BMS MAPs |
| CMN\$\$VAR | Process user options and/or initialize other variables |
| CMN\$PARM | Set global (i.e. shop standard) compile, link, etc. parameters. |
| VAR\$xxxx OPT | Optional include where xxxx is application mnemonic for general application specific staging variables. |
| PRM\$xxxx OPT | Optional include where xxxx is application mnemonic to set application-specific compile, link, etc. parameters. |
| CMN\$\$XSC | Extract source code member from baseline library during recompile or from staging library during staging. |
| CMN\$\$WRT | Parse source code and expand copy structures, including COPY, EXEC SQL, ++INCLUDE and -INC. |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$MPG | Generate BMS MAPs from assembly of MAP source code |
| CMN\$\$MPL | Link BMS MAPs |
| CMN\$\$MPD | Generate BMS MAP DSECTs (copybooks) from MAP source |
| CMN90 | Activate components staged in batch |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN\$\$TRK | Track the success/failure of jobs submitted by generate |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNPLI | Procedure to process PL/I source code |
| CMN\$\$VAR | Process user options and/or initialize other variables |
| CMN\$PARM | Set global (i.e. shop standard) compile, link, etc. parameters. |

| Skeleton | Purpose |
|---------------|--|
| VAR\$xxxx OPT | Optional include where xxxx is application mnemonic for general application specific staging variables. |
| PRM\$xxxx OPT | Optional include where xxxx is application mnemonic to set application-specific compile, link, etc. parameters. |
| CMN\$\$EAR | Ensure nothing has changed since autoretrieve job was generated. |
| CMN\$\$XSC | Extract source code member from baseline library during recompile or from staging library during staging. |
| CMN\$\$WRT | Parse source code and expand copy structures, including COPY, EXEC SQL, ++INCLUDE and -INC. |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$PDB | Perform a DB2 precompile |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN\$\$CIC | Perform a CICS precompile |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$PLI | Compile PL/I source code |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$OBJ | Process object code |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$SSI | Process link-edit control cards |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$LNK | Link edit a program |
| CMN\$\$OBL | Process object code |
| CMN\$\$SYL | Build the SYSLIB link-edit concatenation |
| CMN\$\$ILL | Build the concatenation used during relink |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN\$\$DBC | Copy DB2 DBRM to a DBR staging library |
| CMN\$\$CND | Check for a condition code failure |
| CMN90 | Activate components staged in batch |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |

| Skeleton | Purpose |
|-----------------|--|
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$RMS | Reset PDF statistics in a partitioned dataset |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN\$\$TRK | Track the success/failure of jobs submitted by generate |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNSASC | Procedure to process SAS C source code |
| CMN\$\$VAR | Process user options and/or initialize other variables |
| CMN\$PARAM | Set global (i.e. shop standard) compile, link, etc. parameters. |
| VAR\$xxxx OPT | Optional include where xxxx is application mnemonic for general application specific staging variables. |
| PRM\$xxxx OPT | Optional include where xxxx is application mnemonic to set application-specific compile, link, etc. parameters. |
| CMN\$\$XSC | Extract source code member from baseline library during recompile or from staging library during staging. |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$OBJ | Process object code |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$SSI | Process link-edit control cards |
| CMN\$\$CND | Check for a condition code failure |
| CMN\$\$LNK | Link edit a program |
| CMN\$\$OBL | Process object code |
| CMN\$\$SYL | Build the SYSLIB link-edit concatenation |
| CMN\$\$ILL | Build the concatenation used during relink |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN90 | Activate components staged in batch |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |

| Skeleton | Purpose |
|-----------------|---|
| CMN\$\$TRK | Track the success/failure of jobs submitted by generate |
| CMN\$\$XVP | Send notifications to an IP address and port #. |

Promotion Skeletons

These skeletons are used to build JCL for promotion jobs.

| Skeleton | Purpose |
|-------------------|--|
| CMN\$\$PRM | Perform package promotion or demotion |
| CMN\$\$PSQ | DB2 OPTION: Present SQL/DDDL components to DB2I |
| CMN\$\$PRB | DB2: bind DB2 plans during promotion or demotion |
| CMN\$\$PRO | Copy staging libraries to promotion libraries |
| CMN\$\$CLN | Cleanup prior promotion libraries |
| CMN\$\$DEL | Delete promoted members during demotion |
| CMN\$\$PST | DB2 OPTION: Recycle changed stored procedures |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMNPRMER | Local promotion or demotion job failure |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMNRPMDL | Promotion cleanup at development site |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMN\$\$RPM | Promotion or demotion to REMOTE sites |
| CMN\$\$BDT | Invoke BDT data transmission vehicle |
| CMN\$\$NDM | Invoke Connect:Direct data transmission vehicle at a development site |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN\$\$XCD | Invoke XCOM data Transmission vehicle at development site (skeleton CMN\$\$XCP is used at a remote site) |
| CMN\$\$FTP | Invoke FTP data transmission vehicle |
| CMN\$\$F15 | Invoke FTP data transmission for HFS data |
| CMN\$\$F16 | Invoke FTP data transmission vehicle for z/OS |
| CMNSUBIR | Utility skeleton to handle repeats |

| Skeleton | Purpose |
|-----------------|--|
| CMN\$\$NET | Invoke NetMaster™ data transmission vehicle |
| CMN\$\$SNM | Used during freeze to delimit jobs from one another |
| CMN\$\$RPJ | Build the remote job card |
| CMN\$\$DSN | Define and initialize system dataset names |
| CMN\$\$JBL | Generate the JOBLIB DD concatenation |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNRPMCR | Perform remote promotion or demotion |
| CMN\$\$RCV | Receive XMITted datasets |
| CMNSUBIR | Utility skeleton to handle repeats |
| CMN\$\$RPB | DB2 Remote bind plans during promotion/demotion |
| CMN\$\$JBL | Generate the JOBLIB DD concatenation |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN\$\$PRO | Copy staging libraries to promotion libraries |
| CMN\$\$CLN | Cleanup prior promotion libraries |
| CMN\$\$DEL | Delete promoted members during demotion |
| CMN\$\$RSQ | Present SQL/DDDL components to DB2 |
| CMN\$\$PST | Recycle changed stored procedures and functions |
| CMNRPMT0 | Perform success transmission to local site |
| CMN\$\$BDT | Invoke BDT data transmission vehicle |
| CMN\$\$PND | Execute Connect:Direct data transmission vehicle |
| CMN\$\$XCP | Invoke XCOM data transmission vehicle at a remote site |
| CMN\$\$FTP | Invoke FTP data transmission vehicle |
| CMNRPMT9 | Perform failure transmission to local site |
| CMN\$\$BDT | Invoke BDT data transmission vehicle |
| CMN\$\$PND | Execute Connect:Direct data transmission vehicle |
| CMN\$\$XCP | Invoke XCOM data transmission vehicle at a remote site |
| CMN\$\$FTP | Invoke FTP data transmission vehicle |
| CMNRPMDR | Perform remote promotion cleanup at remote site |
| CMN\$\$SNM | Delimiter to separate the multiple jobs built at freeze time |
| CMN\$\$JCD | Build a job card for all user-initiated jobs |
| CMNRPMB0 | Remote promotion was successful |
| CMN\$\$XVP | Send notifications to an IP address and port #. |

| Skeleton | Purpose |
|-----------------|---|
| CMN\$\$SNM | Used during freeze to delimit jobs from one another |
| CMN\$\$JCD | Build a job card for all user-initiated jobs |
| CMNRPMB9 | Remote promotion failed |
| CMN\$\$XVP | Send notifications to an IP address and port #. |

Installation Skeletons

These skeletons are used to build JCL for install jobs.

| Skeletons | Purpose |
|------------------|--|
| CMN10 | Job to send package &PKGNAME from &DEVNODE to &RSTNDE |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN\$\$BDT | Invoke BDT data transmission vehicle |
| CMN\$\$N00 | Process Connect:Direct for HFS |
| CMN\$\$NDM | Invoke Connect:Direct data transmission vehicle at a development site |
| CMN\$\$N01 | Process Connect:Direct for non-HFS |
| CMN\$\$NDM | Invoke Connect:Direct data transmission vehicle at a development site |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN\$\$X10 | Transfer package flat file, package execution JCL, remote promotion execution JCL and package staging libraries to &RMTSITE via XCOM |
| CMN\$\$F10 | Transfer package flat file, package execution JCL, remote promotion execution JCL and package staging libraries to &RMTSITE via FTP |
| CMN\$\$N10 | Transfer package flat file, package execution JCL, remote promotion execution JCL and package staging libraries to &RMTSITE via NETMASTER |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$X11 | Insert flat file in &RMTSITE package master via XCOM |
| CMN\$\$F11 | Insert flat file in &RMTSITE package master via FTP |
| CMN\$\$N11 | Rebuild staging libraries at &RMTSITE following transmission via NETMASTER |
| CMN10ERR | CMN10 job failure error processing |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN11 | Job to insert &PKGNAME information in package master at &RSTNDE |

| Skeletons | Purpose |
|------------------|--|
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN\$\$JBL | Generate the JOBLIB DD concatenation |
| CMN\$\$SNM | Used during freeze to delimit jobs |
| CMN\$\$DSN | Define and initialize system dataset names |
| CMN\$\$RCV | Receive XMITed datasets |
| CMNSUBIR | Utility skeleton to handle repeats |
| CMN00INS | Communicate with started task |
| CMN11ERR | CMN11 Job failure error processing |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN14 | Job to trigger submission of CMN15 job at &DEVNODE |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN\$\$PND | Execute Connect:Direct data transmission vehicle at a remote site |
| CMN\$\$X15 | Acknowledge distribution via XCOM |
| CMN\$\$F14 | Acknowledge distribution via FTP |
| CMN\$\$N15 | Acknowledge distribution via NETMASTER |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN15 | Job to acknowledge package &PKGNAME sent from &DEVNODE to &RSTNDE |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN17 | Job to insert CMN20 job or CMN31 job into scheduling system |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN\$\$CAS | Insert jobs into CA 7 scheduling system |

| Skeletons | Purpose |
|------------------|--|
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN\$\$ESP | Insert jobs into CA Workload Automation ESP Edition scheduling system |
| CMN\$\$JBT | Insert jobs into CA Jobtrac scheduling system |
| CMN\$\$CTM | Insert jobs into Control-M scheduling system |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN18 | Job to trigger submission of CMN19 job at &DEVNODE |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN\$\$BDT | Invoke BDT data transmission vehicle |
| CMN\$\$PND | Execute Connect:Direct data transmission vehicle at a remote site |
| CMN\$\$X19 | Acknowledge distribution failed via XCOM |
| CMN\$\$F19 | Acknowledge distribution failed via FTP |
| CMN\$\$N19 | Acknowledge distribution failed via NETMASTER |
| CMN19 | Notify package creator that &PKGNAME distribution failed |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN20 | Job to install package &PKGNAME into production |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN\$\$WSI | Stop/start/refresh WebSphere application |
| CMN\$\$SQL | Present SQL/DDI components to DB2 (DB2 Option) |
| CMN\$\$STP | Recycle changed stored procedures (DB2 Option) |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARM DD statement |

| Skeletons | Purpose |
|------------------|--|
| CMN20ERR | CMN20 Job Failure error processing |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN20T | Job to install temporary package &PKGNAME at &RMTSITE |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMNINLPM | Cleanup promotion libraries at sites with no installation process. |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN\$\$CA7 | Delete CA 7 schedule entries |
| CMN20ERR | CMN20 Job Failure error processing |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN21 | DB2: Job to perform DB2 binds for installation of package &PKGNAME into production libraries |
| CMN\$\$JBL | Generate the JOBLIB DD concatenation |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN24 | Job to trigger submission of CMN25 job at &DEVNODE |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |

| Skeletons | Purpose |
|------------------|--|
| CMN\$\$PND | Execute Connect:Direct data transmission vehicle at a remote site |
| CMN\$\$X25 | Acknowledge installation via XCOM |
| CMN\$\$F25 | Acknowledge installation via FTP |
| CMN\$\$N25 | Acknowledge installation via NETMASTER |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN25 | Job to acknowledge package &PKGNAME installation at &DEVNODE |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMNINRPM | Clean up promotion libraries at installed sites |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMNINLPM | Cleanup promotion libraries at sites with no installation process. |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN28 | Job to trigger submission of CMN29 job at &DEVNODE |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN\$\$PND | Execute Connect:Direct data transmission vehicle at a remote site |
| CMN\$\$X29 | Acknowledge installation failed via XCOM |
| CMN\$\$XCP | Invoke XCOM data transmission vehicle at a remote site |
| CMN\$\$F29 | Acknowledge installation failed via FTP |
| CMN\$\$FTP | Invoke FTP data transmission vehicle at a remote site |
| CMN\$\$F15 | Invoke FTP data transmission for HFS data |
| CMN\$\$F16 | Invoke FTP data transmission for z/OS |
| CMN\$\$N29 | Acknowledge installation failed via NETMASTER |
| CMN\$\$NET | Invoke NetMaster data transmission vehicle |

| Skeletons | Purpose |
|------------------|--|
| CMN29 | Job to notify package creator that &PKGNAME installation failed |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN99INS | Notify package creator if any previous step abends (of an install process) and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN30 | Job to perform baseline ripple of package &PKGNAME. See " CMNUPDAT - Stacked Reverse Delta Management " on page 129 for information about the TEXT keyword to the PARM parameter of the EXEC statement that invokes program CMNUPDAT. The TEXT keyword value determines the lines of code that are flagged as changes and are therefore included in the SRD image. |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN\$\$CA7 | Delete CA 7 schedule entries |
| CMN30PAN | Panvalet baseline ripple/delete/rename processing |
| CMN\$\$PLP | Format panvalet and Librarian control cards |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN30LIB | Librarian baseline ripple/delete/rename processing |
| CMN\$\$PLP | Format panvalet and Librarian control cards |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN30SRD | Stacked reverse delta baseline ripple/delete/rename processing |
| CMN30REN | Rename components during baseline ripple |
| CMNPRDIM | Check to see if we really copy to production |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN30PDS | PDS baseline ripple component |
| CMN30CPY | Baseline ripple components using IEBCOPY |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN30REN | Rename components during baseline ripple |
| CMNPRDIM | Check to see if we really copy to production |
| CMN30DEL | Scratch components from baseline |
| CMN30HFS | HFS Delta baseline ripple/delete/rename processing |
| CMNINLPM | Cleanup promotion libraries at sites with no installation process. |
| CMN\$\$SPR | Include SER#PARM DD statement |

| Skeletons | Purpose |
|------------------|--|
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$SLN | Increment list file numbers for compressed listings |
| CMN00INS | Communicate with started task during install |
| CMN30ERR | CMN30 Job Failure error processing |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN31T | Job to cycle (de-install) temporary package &PKGNAME |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN32 | DB2: Job to perform DB2 binds for baseline ripple of package &PKGNAME |
| CMN\$\$JBL | Generate the JOBLIB DD concatenation |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN34T | Job to trigger submission of CMN35T job at &DEVNODE |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN\$\$PND | Execute Connect:Direct data transmission vehicle at a remote site |
| CMN\$\$X35 | Acknowledge temporary package cycled via XCOM |
| CMN\$\$XCP | Invoke XCOM data transmission vehicle at a remote site |
| CMN\$\$F35 | Acknowledge temporary package cycled via FTP |
| CMN\$\$N35 | Acknowledge temporary package cycled via NETMASTER |

| Skeletons | Purpose |
|------------------|--|
| CMN\$\$NET | Invoke NetMaster data transmission vehicle |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN35T | Acknowledge temporary package &PKGNAME has been cycled |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes, as required |
| CMN\$\$DSN | Define and initialize system dataset names |
| CMN\$\$JBL | Generate the JOBLIB DD concatenation |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN37 | Job to maintain the impact analysis dataspace |
| CMN\$\$JNM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$SLN | Increment list file numbers for compressed listings |
| CMN00INS | Communicate with started task during install |
| CMN38T | Job to trigger submission of CMN39T job at &DEVNODE |
| CMN\$\$SJN | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN\$\$PND | Execute Connect:Direct data transmission vehicle at a remote site |
| CMN\$\$X39 | Acknowledge temporary package cycle failed via XCOM |
| CMN\$\$XCP | Invoke XCOM data transmission vehicle at a remote site |
| CMN\$\$F39 | Acknowledge temporary package cycle failed via FTP |
| CMN\$\$N39 | Acknowledge temporary package cycled failed via NetMaster |
| CMN\$\$NET | Invoke NetMaster data transmission vehicle |

| Skeletons | Purpose |
|------------------|--|
| CMN39T | Job to notify package creator the recycle of temporary package &PKGNAME has failed |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN49 | DB2: Job to perform DB2 binds for backout of package &PKGNAME from production libraries |
| CMN\$\$JBL | Generate the JOBLIB DD concatenation |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN50 | Job to back out package &PKGNAME from production libraries |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN\$\$WSB | Stop/start/refresh WebSphere application from the backout process |
| CMN\$\$SQL | DB2 OPTION: Process SQL stored procedure source code |
| CMN\$\$STP | DB2 OPTION: Recycle changed stored procedures |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN50ERR | CMN50 job failure error processing |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |

| Skeletons | Purpose |
|------------------|--|
| CMN50T | Job to back out package &PKGNAME from temporary libraries |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN50ERR | CMN50 job failure error processing |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN51 | Job to send package information dataset for package &PKGNAME from &RSTNDE to &DEVNODE |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN\$\$BDT | Invoke BDT data transmission vehicle |
| CMN\$\$PND | Execute Connect:Direct data transmission vehicle at a remote site |
| CMN\$\$X54 | Acknowledge backout via XCOM |
| CMN\$\$XCP | Invoke XCOM data transmission vehicle at a remote site |
| CMN\$\$F54 | Acknowledge backout via FTP |
| CMN\$\$N54 | Acknowledge backout via NETMASTER |
| CMN\$\$NET | Invoke NetMaster data transmission vehicle |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN54 | Job to trigger submission of CMN55 job at &DEVNODE |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN\$\$RCV | Receive XMITed datasets |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN55 | Job to perform reverse ripple (Backout) of package &PKGNAME |

| Skeletons | Purpose |
|------------------|--|
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN55PAN | Perform reverse ripple of CA Panvalet |
| CMN\$\$PLP | Format CA Panvalet and CA Librarian control cards |
| CMN55LIB | Perform reverse ripple of CA Librarian |
| CMN\$\$PLP | Format CA Panvalet and CA Librarian control cards |
| CMN55SRD | Perform reverse ripple of stacked reverse delta |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN55PDS | Perform reverse ripple of PDS |
| CMN\$\$REN | Rename components during baseline reverse ripple |
| CMN\$\$CPY | Reverse ripple baseline using IEBCOPY |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN55HFS | Perform reverse ripple of HFS |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$SLN | Increment list file numbers for compressed listings |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN55ERR | Communicate with started task during install |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN55T | Job to update package &PKGNAME to backout status |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN56 | DB2: Job to perform DB2 binds for reverse baseline ripple (BACKOUT) of package &PKGNAME |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN\$\$JBL | Generate the JOBLIB DD concatenation |
| CMN\$\$D2X | Build the DB2 system exit load library dsn from the DB2 system load library dsn specified in &db2dsnld |
| CMN00INS | Communicate with started task during install |

| Skeletons | Purpose |
|------------------|--|
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN57 | Perform reverse ripple of impact analysis tables |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$SLN | Increment list file numbers for compressed listings |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN58 | Job to trigger submission of CMN59 job at &DEVNODE |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN\$\$PND | Execute Connect:Direct data transmission vehicle at a remote site |
| CMN\$\$X59 | Acknowledge backout failed via XCOM |
| CMN\$\$XCP | Invoke XCOM data transmission vehicle at a remote site |
| CMN\$\$F59 | Acknowledge backout failed via FTP |
| CMN\$\$FTP | Invoke FTP data transmission vehicle at a remote site |
| CMN\$\$N59 | Acknowledge backout failed via NETMASTER |
| CMN\$\$NET | Invoke NetMaster data transmission vehicle |
| CMN59 | Job to notify package creator of package &PKGNAME backout failure |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN64 | Job to trigger submission of CMN65 job at &DEVNODE |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |

| Skeletons | Purpose |
|------------------|--|
| CMN\$\$BDT | Invoke BDT data transmission vehicle |
| CMN\$\$PND | Execute Connect:Direct data transmission vehicle at a remote site |
| CMN\$\$X65 | Acknowledge revert via XCOM |
| CMN\$\$XCP | Invoke XCOM data transmission vehicle at a remote site |
| CMN\$\$F65 | Acknowledge revert via FTP |
| CMN\$\$FTP | Invoke FTP data transmission vehicle at a remote site |
| CMN\$\$N65 | Acknowledge revert via NETMASTER |
| CMN\$\$NET | Invoke NetMaster data transmission vehicle |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN65 | Job to revert package &PKGNAME at &DEVNODE |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN\$\$RCV | Receive XMITed datasets |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN90 | Activate components staged in batch |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |

Utility And Other Skeletons

These skeletons are used to build JCL for utility jobs and other batch jobs.

| Skeleton | Purpose |
|-----------------|---|
| CMN\$IAX | Job to extract impact analysis data. |
| CMN\$JCD | Build a job card for all user-initiated jobs |
| CMN\$AUD | Job to AUDIT a package |
| CMN\$SPR | Include SER#PARM DD statement |
| CMN\$XVP | Communicate with another platform |
| CMN\$BBL | Job to print or copy baseline / promotion library components |
| CMN\$CBL | Copy baseline / promotion components to a user dataset |
| CMN\$SPR | Include SER#PARM DD statement |
| CMN\$PBL | Print baseline / promotion library components |
| CMN\$SPR | Include SER#PARM DD statement |
| CMN\$XVP | Communicate with another platform |
| CMN\$BLA | Generate Baseline Analyzer Report |
| CMN\$JCD | Build a job card for all user-initiated jobs |
| CMN\$SPR | Include SER#PARM DD statement |
| CMN\$XVP | Send notifications to an IP address and port #. |
| CMN\$BLG | Print the activity log |
| CMN\$JCD | Build a job card for all user-initiated jobs |
| CMN\$XVP | Send notifications to an IP address and port #. |
| CMN\$BRT | Retrieve package components in batch mode from ERO area libraries |
| CMN\$JCD | Build a job card for all user-initiated jobs |
| CMN\$SPR | Include SER#PARM DD statement |
| CMN\$XVP | Send notifications to an IP address and port #. |
| CMN\$BSN | Scan libraries for character strings |
| CMN\$SPR | Include SER#PARM DD statement |
| CMN\$XVP | Send notifications to an IP address and port #. |
| CMN\$CPR | Job to compare staging libraries to baseline or promotion |
| CMN\$LDT | Generate Link Date Report |
| CMN\$JCD | Build a job card for all user-initiated jobs |
| CMN\$SPR | Include SER#PARM DD statement |
| CMN\$NOT | Process the global notification outlist dataset |
| CMN\$XVP | Send notifications to an IP address and port #. |
| CMN\$PRS | Job to compress package staging libraries |

| Skeleton | Purpose |
|-------------------|--|
| CMN\$\$JCD | Build a job card for all user-initiated jobs |
| CMN\$\$PRT | Job to print the comparison listing following an edit session |
| CMN\$\$RPT | Generate ChangeMan ZMF SAS reports |
| CMN\$\$JCD | Build a job card for all user-initiated jobs |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNGRANT | DB2: procedure to grant access to DB2 plans |
| CMN90 | Activate components staged in batch |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNXMLB | Invoke batch XML services |
| CMN\$\$SPR | Include SER#PARAM DD statement |

IMS Option Skeletons

These skeletons are used to build JCL for jobs submitted for the ChangeMan ZMF IMS Option.

IMS Stage Skeletons

These skeletons are used to build JCL for IMS Option stage jobs.

| Skeleton | Purpose |
|-----------------|---|
| CMNDBDGN | IMS Option: procedure to process DBD source code |
| CMN\$\$VAR | Process user options and/or initialize other variables |
| CMN\$PARAM | Set global (i.e. shop standard) compile, link, etc. parameters. |
| VAR\$xxxx OPT | Optional include where xxxx is application mnemonic for general application specific staging variables. |

| Skeleton | Purpose |
|-----------------|--|
| PRM\$xxxx OPT | Optional include where xxxx is application mnemonic to set application-specific compile, link, etc. parameters. |
| CMN\$\$WRT | Parse source code and expand copy structures, including COPY, EXEC SQL, ++INCLUDE and -INC. |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$PDB | Perform a DB2 precompile |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$CIC | Perform a CICS precompile |
| CMN\$\$ASM | |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$OBJ | Process object code |
| CMN\$\$SSI | Process link-edit control cards |
| CMN\$\$LNK | Link edit a program |
| CMN\$\$OBL | Process object code |
| CMN\$\$SYL | Build the SYSLIB link-edit concatenation |
| CMN\$\$ILL | Build the concatenation used during relink |
| CMN\$\$IMS | IMS Option: Parse DBD/PSB source and build ACB statement |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$DBC | Copy DB2 DBRM to a DBR staging library |
| CMN90 | Activate components staged in batch |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$RMS | Reset PDF statistics in a partitioned dataset |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN\$\$TRK | Track the success/failure of jobs submitted by generate |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNMFSGN | IMS Option: Process MFS source code |
| CMN\$\$VAR | Process user options and/or initialize other variables |
| CMN\$PARAM | Set global (i.e. shop standard) compile, link, etc. parameters. |

| Skeleton | Purpose |
|-----------------|--|
| VAR\$xxxx OPT | Optional include where xxxx is application mnemonic for general application specific staging variables. |
| PRM\$xxxx OPT | Optional include where xxxx is application mnemonic to set application-specific compile, link, etc. parameters. |
| CMN\$\$WRT | Parse source code and expand copy structures, including COPY, EXEC SQL, ++INCLUDE and -INC. |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$MFS | Assemble IMS MFS (message format service) source code |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN90 | Activate components staged in batch |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN\$\$TRK | Track the success/failure of jobs submitted by generate |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNPSBGN | IMS Option: Procedure to process program specification block (PSB) |
| CMN\$\$VAR | Process user options and/or initialize other variables |
| CMN\$PARAM | Set global (i.e. shop standard) compile, link, etc. parameters. |
| VAR\$xxxx OPT | Optional include where xxxx is application mnemonic for general application specific staging variables. |
| PRM\$xxxx OPT | Optional include where xxxx is application mnemonic to set application-specific compile, link, etc. parameters. |
| CMN\$\$WRT | Parse source code and expand copy structures, including COPY, EXEC SQL, ++INCLUDE and -INC. |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$SYC | Build the SYSLIB compile concatenation |
| CMN\$\$PDB | Perform a DB2 precompile |
| CMN\$\$CIC | Perform a CICS precompile |
| CMN\$\$ASM | Translate ASSEMBLER source code |
| CMN\$\$OBJ | Process object code |

| Skeleton | Purpose |
|-----------------|--|
| CMN\$\$SSI | Process link-edit control cards |
| CMN\$\$LNK | Link edit a program |
| CMN\$\$OBL | Process object code |
| CMN\$\$SYL | Build the SYSLIB link-edit concatenation |
| CMN\$\$ILL | Build the concatenation used during relink |
| CMN\$\$IMS | IMS Option: Parse DBD/PSB source and build ACB statement |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$DBC | Copy DB2 DBRM to a DBR staging library |
| CMN90 | Activate components staged in batch |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$RMS | Reset PDF statistics in a partitioned dataset |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN\$\$TRK | Track the success/failure of jobs submitted by generate |
| CMN\$\$XVP | Send notifications to an IP address and port #. |

IMS Promotion Skeletons

These skeletons are used to build JCL for IMS Option promotion jobs.

| Skeleton | Purpose |
|-----------------|--|
| CMNIMPRM | IMS Option: Perform package promotion or demotion with IMS option For local sites |
| CMN\$\$PSQ | DB2 Option: Present SQL/DDI components to DB2 |
| CMN\$\$PRB | Bind DB2 plans during promotion or demotion |
| CMNPRMIM | IMS Option: Check if we really copy it to promotion |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMNJSPRM | Submit to the internal reader when there are more than 255 steps during a promotion job |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARM DD statement |

| Skeleton | Purpose |
|------------|--|
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN\$\$JCD | Build a job card for all user-initiated jobs |
| CMNIMPRO | IMS Option: Copy from staging to desired promotion level |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN\$\$IGN | IMS Option: Perform IMS gens at promote or install time |
| CMNJSPRM | Submit to the internal reader when there are more than 255 steps during a remote promotion job |
| CMNPDPGN | IMS Option: Process PSB/DBD source code at promote time |
| CMNMFSPG | IMS Option: Process MFS source code at promotion time |
| CMN\$\$IGN | IMS Option: Perform IMS GENS at promote or install time |
| CMNPDPEX | IMS Option: Process PSB/DBD source code at promote time for expansion before overrides |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMNPDPOV | IMS Option: Process PSB/DBD source code at promote time for overrides to the PSB/DBD source code |
| CMNIMCPY | IMS Option: Synchronize promotion libraries and IMS libraries |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMNIMSPR | IMS Option: Synchronize promotion libraries |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMNJSPRM | Initiate promotion when count > 255 |
| CMN\$\$ACB | IMS Option: Perform an ACB gen |
| CMN\$\$CLN | Cleanup prior promotion libraries |
| CMN\$\$DEL | Delete promoted members during demotion |
| CMN\$\$PRB | DB2: Bind DB2 plans during promotion or demotion |
| CMN\$\$PST | DB2 Option: Recycle changed stored procedures |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARM DD statement |

| Skeleton | Purpose |
|-----------------|--|
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMNRPMDL | Promotion cleanup at development site |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNIMRPM | IMS Option: Main driver for remote promotion |
| CMN\$\$BDT | Invoke BDT data transmission vehicle |
| CMN\$\$NDM | Invoke Connect:Direct data transmission vehicle at a development site |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN\$\$XCD | Invoke XCOM data Transmission vehicle at development site (skeleton CMN\$\$XCP is used at a remote site) |
| CMN\$\$FTP | Invoke FTP data transmission vehicle |
| CMN\$\$NET | Invoke NETMASTER data transmission vehicle |
| CMN\$\$SNM | Used during freeze to delimit jobs from one another |
| CMN\$\$RPJ | Build the remote job card |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMNRPICR | IMS Option: Perform remote promotion or demotion with IMS |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMN\$\$SNM | Used during freeze to delimit jobs from one another |
| CMN\$\$JCD | Build a job card for all user-initiated jobs |
| CMNRPMB0 | Communicate with started task. |
| CMN\$\$XVP | Send notifications to an IP address and port #. |
| CMN\$\$SNM | Used during freeze to delimit jobs from one another |
| CMN\$\$JCD | Build a job card for all user-initiated jobs |
| CMNRPMB9 | Communicate with started task. |
| CMN\$\$XVP | Send notifications to an IP address and port #. |

IMS Installation Skeletons

These skeletons are used to build JCL for IMS Option install jobs.

| Skeletons | Purpose |
|------------------|---|
| CMN20I | IMS Option: Job to install package &PKGNAME into production |
| CMN\$\$SJN | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN00INS | Communicate with started task during install |

| Skeletons | Purpose |
|------------------|--|
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMNJSPRD | Submit to the Internal Reader when there are more than 255 steps during a Package install to Production |
| CMNINIGN | IMS Option: Perform IMS GENS at install time |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMNIDPGN | IMS Option: Process PSB/DBD source code at install time |
| CMNMFSIG | IMS Option: Process MFS source code at install time |
| CMNIDPEX | IMS Option: Process PSB/DBD source code at install time for expansion before overrides |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMNIDPOV | IMS Option: Process PSB/DBD source code at install time for overrides to the PSB/DBD source code |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMNPRDIM | IMS Option: Synchronize production libraries and IMS libraries |
| CMNIMSPD | IMS OPTION: Check if we really copy to production |
| CMNINACB | IMS Option: Perform an ACB gen |
| CMN\$\$SQL | Present SQL/DDDL components to DB2 (DB2 Option) |
| CMN\$\$STP | Recycle changed stored procedures (DB2 Option) |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN20TI | IMS Option: Job to install a temporary package |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |

| Skeletons | Purpose |
|------------------|---|
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMNINLPM | Cleanup promotion libraries at sites with no installation process |
| CMNPRDIM | IMS Option: Check to see if we really copy to production |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |
| CMNINIGN | IMS Option: Perform IMS gens at install time |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMNIDPGN | IMS Option: Process PSB/DBD source code at install time |
| CMNMFSIG | IMS Option: Process MFS source code at install time |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |
| CMNIDPEX | IMS Option: Process PSB/DBD source code at install time for expansion before overrides |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMNIDPOV | IMS Option: Process PSB/DBD source code at install time for overrides to the PSB/DBD source code |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |
| CMNPRDIM | IMS OPTION: Check if we really copy to production |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |
| CMNIMSTP | IMS OPTION: Sync IMS and temporary libraries |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |

| Skeletons | Purpose |
|------------------|--|
| CMNINACB | IMS Option: Perform an ACB gen |
| CMN\$\$CA7 | Delete CA7 schedule entries |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN30I | IMS Option: Job to perform baseline ripple of package &PKGNAME |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN\$\$CA7 | Delete CA7 schedule entries |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMN\$\$SPR | Include SER#PARM DD statement |
| CMN30PAN | CA Panvalet baseline RIPPLE/DELETE/RENAME processing |
| CMN\$\$PLP | Format CA Panvalet and CA Librarian control cards |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMN30LIB | CA Librarian baseline RIPPLE/DELETE/RENAME processing |
| CMN\$\$PLP | Format CA Panvalet and CA Librarian control cards |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMN30CDT | Cascade delta baseline RIPPLE/DELETE/RENAME processing |
| CMN30CPI | IMS Option: Baseline ripple components using IEBCOPY |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN30REN | Rename components during baseline ripple |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN30DEL | Scratch components from baseline |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |

| Skeletons | Purpose |
|------------|--|
| CMN30SRD | Stacked reverse delta baseline RIPPLE/DELETE/RENAME processing. See "CMNUPDAT - Stacked Reverse Delta Management" on page 129 for information about the TEXT keyword to the PARM parameter of the EXEC statement that invokes program CMNUPDAT. The TEXT keyword value determines the lines of code that are flagged as changes and are therefore included in the SRD image. |
| CMN30REN | Rename components during baseline ripple |
| CMN30DEL | Scratch components from baseline |
| CMNPRDIM | IMS Option: Check to see if we really copy to production |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMN30PDS | PDS baseline RIPPLE/RENAME/DELETE processing |
| CMN30CPI | IMS Option: Baseline ripple components using IEBCOPY |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN30REN | Rename components during baseline ripple |
| CMNPRDIM | IMS Option: Check to see if we really copy to production |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN30DEL | Scratch components from baseline |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMNINIGN | IMS Option: Perform IMS GENS at install time |
| CMNJSPRD | Submit to the internal reader when there are more than 255 steps during a package install to production |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMNIDPGN | IMS Option: Process PSB/DBD source code at install time |
| CMNMFSIG | IMS Option: Process MFS source code at install time |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMNIDPEX | IMS Option: Process PSB/DBD source code at install time for expansion before overrides |
| CMN\$\$SPR | Include SER#PARAM DD statement |

| Skeletons | Purpose |
|------------------|--|
| CMNIDPOV | IMS Option: Process PSB/DBD source code at install time for overrides to the PSB/DBD source code |
| CMNPRDIM | IMS Option: Check to see if we really copy to production |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMNIMSBL | Synchronize baseline libraries and IMS libraries |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMNINLPM | Cleanup promotion libraries at sites with no installation process |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMNJSBAS | Submit to the internal reader when there are more than 255 steps during a baseline ripple |
| CMNINACB | IMS Option: Perform an ACB gen |
| CMN\$\$SLN | Increment list file numbers for compressed listings |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN31TI | IMS Option: Job to cycle (de-install) temporary package &PKGNAME |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN50I | IMS Option: Job to back out package &PKGNAME from production |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN\$\$SQL | Present SQL/DDDL components to DB2 (DB2 Option) |
| CMN\$\$STP | Recycle changed stored procedures (DB2 Option) |
| CMN00INS | Communicate with started task during install |

| Skeletons | Purpose |
|------------------|--|
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99INS | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN50TI | IMS Option: Job to back out package &PKGNAME from temp libraries |
| CMN\$\$SJM | Used at freeze time to build the job cards for production library and/or remote site processes. |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8. |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN55I | IMS Option: Job to perform reverse ripple of package &PKGNAME |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN55PAN | Perform reverse ripple of CA Panvalet |
| CMN\$\$PLP | Format CA Panvalet and CA Librarian control cards |
| CMN55LIB | Perform reverse ripple of CA Librarian |
| CMN\$\$PLP | Format CA Panvalet and CA Librarian control cards |
| CMN55CDT | Perform reverse ripple of cascade delta |
| CMN55CPI | IMS Option: Reverse ripple baseline using IEBCOPY |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN55REN | Rename components during baseline reverse ripple |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN55SRD | Perform reverse ripple of stacked reverse delta |
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN55PDS | Perform reverse ripple of PDS |
| CMN55REN | Rename components during baseline reverse ripple |
| CMN55CPI | IMS Option: Reverse ripple baseline using IEBCOPY |

| Skeletons | Purpose |
|------------------|--|
| CMN\$\$ENQ | Single thread ChangeMan ZMF jobs |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$SLN | Increment list file numbers for compressed listings |
| CMN00INS | Communicate with started task during install |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| | Notify package creator if any previous step (of an install process) abends and/or the return code of any previous step is greater than or equal to 8 |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |
| CMN55TI | IMS Option: Job to update package &PKGNAME to backout status |
| CMN\$\$JNM | Used at freeze time to build the job cards for baseline ripple and/or development site processes. |
| CMN00 | Communicate with started task |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN99 | Notify package creator if any previous step abends and/or the return code of any previous step is greater than or equal to 8. |
| CMN\$\$SPR | Include SER#PARAM DD statement |
| CMN\$\$PCP | Merge all SYSPRINT datasets into a single file and/or create a compressed listing in the LST staging library |

IMS Sample Skeletons

These are sample skeletons for IMS install and promotion functions.

| Skeleton | Purpose |
|-----------------|--|
| CMNINASW | IMS Option: Sample ACB library swap for production / install |
| CMNINFSW | IMS Option: Sample format library swap for production / install |
| CMNINIBK | IMS Option: Sample IMS library backup for install or baseline |
| CMNINIRB | IMS Option: Sample IMS library recovery for install or baseline |
| CMNPRASW | IMS Option: Sample ACB library swap for remote / local promotion |
| CMNPRFSW | IMS Option: Sample format library swap for remote / local promotion |
| CMNPRIBK | IMS Option: Sample IMS library backup for remote / local promotion |
| CMNPRIRB | IMS Option: Sample IMS library recovery for remote / local promotion |

Index

A

abend
 CMNSSIDN
 CMNSSIDN
 abend 128
Adobe Acrobat 12
auditing
 CMNWRITE 143

B

baseline
 ripple 129

C

CEDA language review 112
Checkout Options panel (CMNMCKOT) 72, 73,
 74, 76
CICS CSD extract 106
CMNBAHST 82
CMNBAT90 83, 94
CMNCICS6 106
CMNEX001 47
CMNEX002 47
CMNEX003 47
CMNEX004 48
CMNEX005 48
CMNEX006 48
CMNEX007 48
CMNEX008 48
CMNEX009 49
CMNEX010 50
CMNEX011 50
CMNEX012 50
CMNEX014 50
CMNEX015 51
CMNEX019 51
CMNEX020 52
CMNEX021 52
CMNEX022 52
CMNEX023 52
CMNEX024 53
CMNEX025 53
CMNEX026 53
CMNEX027 53

CMNEX028 54
CMNEX030 54
CMNEX031 54
CMNEX032 54
CMNEX033 54
CMNEX034 55
CMNEX035 55
CMNEX036 55
CMNEX037 55
CMNEX038 56
CMNEX039 56
CMNEX040 56
CMNEX041 56
CMNEX042 56
CMNEX043 57
CMNEX101 57
CMNEX102 57
CMNEX103 57
CMNEX201 58
CMNEXINS 46
CMNFIX10
 abend codes 114
CMNFIXMN 112
CMNIALD0 115
CMNIALU0 115
CMNPML0D 117
CMNSSIDN 123
CMNUPDAT 129
CMNWRITE 135
 DD statements 136
compressed listings
 browsing 151
COPY 142
COPY management 135
copy utility 144
customizing ChangeMan ZMF 15

D

DB2
 impact analysis 115
DD statements
 CMNWRITE 136
 SERCOPY 145

E

exits
 user 39
extract
 masterfile 117

H

history initial 82

I

imbedding skeletons 22
impact analysis
 DB2 115
Import option 108
INCLUDE 142
INCLUDE management 135

L

language
 CEDA review 112
link edit control 123

M

masterfile
 extract 117

N

naming conventions 20

O

options
 SYSIN 87, 126

P

panels
 Checkout Options (CMNMCKOT) 72, 73, 74,
 76
parameters
 SYSIN 102, 137

R

refreshing VLA and LLA 44
reports
 customizing 154–163
 requesting a report online 154
 submitting a batch job 155
 summary 142
REPORTS member of CNTL library 155
REXX program library 154
ripple baseline 129

S

SER#PARM DD statement 155
SERABEND DD statement 155
SERCOPY 144
 DD statements 145
SEREX001 45
SETSSI generate 112
skeleton variables 21
skeletons
 imbedding 22
 with file tailoring 24
source load relationships 83, 94
statements
 COPY 142
 INCLUDE 142
syntax checking
 with file tailoring 27
SYSABEND DD statement 155
SYSEXEC DD concatenation 155
SYSIN options 87, 126
SYSIN parameters 102, 137
SYSTSPRT DD statement 155

U

user exits
 CMNEX001 47
 CMNEX002 47
 CMNEX003 47
 CMNEX004 48
 CMNEX005 48
 CMNEX006 48
 CMNEX007 48
 CMNEX008 48
 CMNEX009 49
 CMNEX010 50
 CMNEX011 50
 CMNEX012 50
 CMNEX014 50
 CMNEX015 51
 CMNEX019 51

- CMNEX020 52
- CMNEX021 52
- CMNEX022 52
- CMNEX023 52
- CMNEX024 53
- CMNEX025 53
- CMNEX026 53
- CMNEX027 53
- CMNEX028 54
- CMNEX030 54
- CMNEX031 54
- CMNEX032 54
- CMNEX033 54
- CMNEX034 55
- CMNEX035 55
- CMNEX036 55
- CMNEX037 55
- CMNEX038 56
- CMNEX039 56
- CMNEX040 56
- CMNEX041 56
- CMNEX042 56
- CMNEX043 57
- CMNEX101 57
- CMNEX102 57
- CMNEX103 57
- CMNEX201 58
- CMNEXINS 46
 - defined 39
- SEREX001 45
- utilities 81
 - CMNWRITE 135
 - SERCOPY 144
- utility
 - CMNBAHST 82
 - CMNBAT90 83, 94
 - CMNCICS6 106
 - CMNFXMN 112
 - CMNIALD0 115
 - CMNIALU0 115
 - CMNPML0D 117
 - CMNSSIDN 123
 - CMNUPDAT 129
 - SERCOPY 144

X

- XML services
 - used in reporting 161–162

