MICRO FOCUS | SERENA

ChangeMan® ZMF

Java / zFS Getting Started Guide

Product version: 8.2
Publication date: September 2018

Table of Contents

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.

Audience and scope

This document is an addendum to the ChangeMan ZMF documentation set. It provides information that is necessary to use ZMF to manage application components stored in USS file systems, especially Java application components.

This document is intended for ChangeMan ZMF installers, global administrators, application administrators, and security administrators. It is also intended for developers who use ChangeMan ZMF to manage changes to applications with components stored in USS file systems.

Navigating this book

This manual is organized as follows:

- Chapter 1 provides an introduction to USS file support in ChangeMan ZMF.
- Chapter 2 describes how to work with USS file directories and file names in the ISPF interface to ChangeMan ZMF.
- Chapter 3 lists the steps required to configure ChangeMan ZMF to manage Java components and other zFS files.
- Chapter 4 tells developers how to use ChangeMan ZMF to work with Java components and applications.

Change Bars

Change bars in the left margin identify substantive changes in this publication since the last published version.

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
Administrator's Guide	Describes ChangeMan ZMF features and functions with instructions for choosing options and configuring global and application administration parameters.
ChangeMan ZMF Quick Reference	Provides a summary of the commands you use to perform the major functions in the ChangeMan ZMF package life cycle.

Manual	Description
Customization Guide	Provides information about ChangeMan ZMF skeletons, exits, and utility programs that will help you to customize the base product to fit your needs.
DB2 Option Getting Started Guide	Describes how to install and use the DB2 Option of ChangeMan ZMF to manage changes to DB2 components.
ERO Concepts	Discusses the concepts of the ERO Option of ChangeMan ZMF for managing releases containing change packages.
ERO Getting Started Guide	Explains how to install and use the ERO Option of ChangeMan ZMF to manage releases containing change packages.
ERO Messages	Describes system messages and codes produced by ChangeMan ZMF ERO.
ERO XML Services User's Guide	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.
High-Level Language Functional Exits Getting Started Guide	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.
IMS Option Getting Started Guide	Provides instructions for implementing and using the IMS Option of ChangeMan ZMF to manage changes to IMS components.
INFO Option Getting Started Guide	Describes two methods by which ChangeMan ZMF can communicate with other applications: ■ Through a VSAM interface file. ■ Through the Tivoli Information Management for z/OS product from IBM.
Installation Guide	Provides step-by-step instructions for initial installation of ChangeMan ZMF. Assumes that no prior version is installed or that the installation will overlay the existing version.
Java / zFS Getting Started Guide	Provides information about using ZMF to manage application components stored in USS file systems, especially Java application components.
Load Balancing Option Getting Started Guide	Explains how to install and use the Load Balancing Option of ChangeMan ZMF to connect to a ChangeMan ZMF instance from another CPU or MVS image.
M+R Getting Started Guide	Explains how to install and use the M+R Option of ChangeMan ZMF to consolidate multiple versions of source code and other text components.
M+R Quick Reference	Provides a summary of M+R Option commands in a handy pamphlet format.
Messages	Explains messages issued by ChangeMan ZMF, SERNET, and System Software Manager (SSM) used for the Staging Versions feature of ChangeMan ZMF.

Manual	Description
ChangeMan ZMF Migration Guide	Gives guidance for upgrading ChangeMan ZMF.
OFM Getting Started Guide	Explains how to install and use the Online Forms Manager (OFM) option of ChangeMan ZMF.
SER10TY User's Guide	Gives instructions for applying licenses to enable ChangeMan ZMF and its selectable options.
User's Guide	Describes how to use ChangeMan ZMF features and functions to manage changes to application components.
XML Services User's Guide	Documents the most commonly used features of the XML 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 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 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
italics	Introduces new terms that you may not be familiar with and occasionally indicates emphasis.
bold	Indicates panel titles, field names, and emphasizes important information.
UPPERCASE	Indicates keys or key combinations that you can use. For example, press ENTER.
`monospace`	Indicates syntax examples, values that you specify, or results that you receive.
`monospaced italics`	Indicates names that are placeholders for values you specify; for example, *`filename`*.
`monospace bold`	Indicates the results of an executed command.
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.

Chapter 1
Introduction

ChangeMan ZMF includes support for Java applications, providing:

- Secure, rules based management of Java application artifacts in ChangeMan package life cycle
- Stability, reliability, and security of the mainframe platform
- ZMF for Eclipse and RDz user interfaces.

Java Support

Java application components that are managed by ChangeMan ZMF include:

- Java source
- Java class
- JAR build control
- JAR
- WAR build control
- WAR
- Java application HTML
- Other Java application artifacts

Java build processes in ChangeMan ZMF create the following:

- Java class
- JAR
- WAR

zFS in UNIX System Services under z/OS

Support for Java includes support for zFS files under UNIX System Services in z/OS. ChangeMan ZMF supports zFS files with names up to 255 characters and zFS path names up to 1023 characters long.

zFS directory paths and file are used for Java components in place of PDS(E) libraries for the following:

- Staging
- Promotion
- Production
- Baseline

User Interface

While the ISPF interface to ChangeMan ZMF is modified to accommodate the long zFS path and component names required for Java, most Java developers will be unfamiliar with the mainframe environment and will not be efficient working through the ISPF interface.

ChangeMan ZMF works with IDEs like Eclipse and RDz that provide a more familiar environment for Java developers. Ask your account representative about the ChangeMan ZMF Client Pack, which includes ChangeMan for Eclipse, an integration plug-in for both IBM Rational Developer and the open-source Eclipse IDE.

zFS Limitations

There are some limitations on the use of zFS in ChangeMan ZMF 8.1.

No zFS In Converted Packages

If you upgrade from a version before 7.1, you cannot check out or stage a zFS component into a package that was created before you upgraded.

The package records created in an earlier ZMF version do not contain the zFS staging model data set names that you enter in global administration after you convert the package master from an earlier version.

You can only check out or stage zFS components into packages created in ChangeMan ZMF 7.1 or later.

Functions Without zFS Support

These functions do not support zFS files in ChangeMan ZMF:

- Scan Baseline For Character Strings (=1.S) - short message 'NOT SUPPORTED' and Long message: 'CMN6503I - Scan of this baseline storage means is not supported.'
- Scan sub-function of Browse\Print\Copy Baseline Or Promotion (=1.B)
- M+R Option (=C)

Previously this was not available but now this function does support zFS files in ChangeMan ZMF:

- Compare Staging To Baseline Or Promotion (=1.C)

Chapter 2

Working With zFS In ZMF

ChangeMan ZMF includes support for Java applications, providing:

- Secure, rules based management of Java application artifacts in ChangeMan package life cycle
- Stability, reliability, and security of the mainframe platform
- ZMF for Eclipse and RDz user interfaces.

Working with Long Fields in ISPF

Component names in USS file systems can be 256 characters long, and path names can be 1024 characters long.

In the ChangeMan ZMF ISPF interface, there are three ways users can see a long component or path name on a panel whose total width is limited to 80 characters.

- Scroll in the panel field
- Zoom in on the field with EXPAND
- Display an alternate panel

In this section, example package ACTP000052 contains these components:

Staging Library (Path) Name	Component Name
/cmntp/s6/ACTP/#000052/d/JVS	hellow.java
/cmntp/s6/ACTP/#000052/d/JCT	hellow.jct
/cmntp/s6/ACTP/#000052/d/JVS	org/dom/xpath/jhfhth30long.java
/cmntp/s6/ACTP/#000052/d/HTH	org/jdom/xpath/jhfhth30long.hth
/cmntp/s6/ACTP/#000052/d/HTH	testfile.hth

IMPORTANT! In USS file systems, a "component name" may include a partial directory path because the actual name of the file is unique only within a hierarchy that includes it.

When line command **S2** is entered by package ACTP000052 on the **Change Package List** panel, the **Stage: *package* Components** panel (CMNSTG01) is displayed in this format.

```
CMNSTG01                  STAGE: ACTP000052 Components         Row 1 to 5 of 5
Command ===> _____________________________________________ Scroll ===> CSR

   Name             + Type Status   Changed         Procname User     Request
__ hellow.java        JVS  ACTIVE   20150305 145828 CMNJAVA  USER015
__ hellow.jct         JCT  ACTIVE   20150309 184115 CMNJAR   USER015
__ org/dom/xpath/jhfh JVS  ACTIVE   20150309 200446 CMNJAVA  USER015
__ org/jdom/xpath/jhf HTH  ACTIVE   20150309 192827          USER015
__ testfile.hth       HTH  ACTIVE   20150309 192103          USER015
****************************** Bottom of data ********************************
```

When a data field is *longer* than the panel field used to display it, a **+** (plus) is shown to the right of the panel field. If the panel field is in a list, the + is shown over the panel field column, in the heading, as shown in the example above.

Notice that the names of the first two components and the last component fit in the **Name** column field. However, the names of the third and fourth components appear to overflow the Name field.

Scrolling LEFT and RIGHT

To see more of a long name that is truncated by a short panel field, you can scroll to the right, and then scroll back to the left.

- To scroll to the *right* in a field, place your cursor in the field and press **PF11**. (You can also type **RIGHT** in the **Command** line, place your cursor in the field, and press **ENTER**.)
- To scroll to the *left* in a field, place your cursor in the field and press **PF10**. (You can also type **LEFT** in the **Command** line, place your cursor in the field, and press **ENTER**.)

This panel shows the **Name** field after scrolling right one time.

```
CMNSTG01                  STAGE: ACTP000052 Components         Row 1 to 5 of 5
Command ===> _____________________________________________ Scroll ===> CSR

   Name            -+ Type Status   Changed          Procname User     Request
__                    JVS  ACTIVE   20150305 145828 CMNJAVA  USER015
__                    JCT  ACTIVE   20150309 184115 CMNJAR   USER015
__ fhth30long.java    JVS  ACTIVE   20150309 200446 CMNJAVA  USER015
__ hfhth30long.hth    HTH  ACTIVE   20150309 192827          USER015
__                    HTH  ACTIVE   20150309 192103          USER015
****************************** Bottom of data ********************************
```

Rules for scrolling in long panel fields:

- The **SCROLL** amount at the upper right of the panel determines the scroll amount for long panel fields.
- You might have to scroll right more than once to see the end of a long field.
- When you have scrolled to right end of a long field, a **-** (minus) replaces the **+** (plus) to indicate that you can only scroll left from that point
- Both -+ (minus plus) are displayed when you can scroll both right and left from your current position in a long field.
- If you type **M** (MAXIMUM) on the **Command** line before you position your cursor in a long field, when you press **PF11** or **PF10** the field scrolls all the way to the end or the beginning respectively.

Long Field Zoom - EXPAND

Rather than pressing **PF11** or **PF10** multiple times to scroll through a long field, you can zoom in on the field by placing your cursor over the field and pressing **PF4** to execute the ISPF **EXPAND** command. EXPAND displays the entire field in a pop-up panel.

If you place the cursor on the **NAME** field of the fourth component listed on the STAGE: package COMPONENTS panel above and you press **PF4**, the pop-up **CMPNAME+0** panel (ISPEXPND) is displayed.

```
+-------------------------------- CMPNAME+0 ---------------------------------+
| ISPEXPND                                                 Line   1 of      4 |
| Command ===> _____________________________________________ Scroll ===> PAGE   |
|                                                                              |
| org/jdom/xpath/jhfhth30long.hth                                              |
|                                                                              |
...
|                                                                              |
|                                                                              |
+------------------------------------------------------------------------------+
```

Zoom panel rules:

- The pop-up panel has the same attributes as the original panel field. If the original panel field is display-only, then the pop-up panel is also display-only. If the original panel field is available for input, you can type on the pop-up panel.
- If the original panel field is available for input, you can type up to 256 characters on multiple pop-up panel lines for a component name, or 1024 characters on multiple lines for a path name.
- Press **PF3** to exit the pop-up long name panel and return to the original panel.

NOTE Most ZMF panels in the ISPF interface follow this rule: "Press ENTER to process; Enter END or CANCEL command to exit." However, the pop-up panel for expanded long name fields requires END or PF3 to save data entered or changed on the panel.

Clearing Long Names From Panel Input Fields

If a long name extends beyond the visible end of an input field on a ChangeMan ZMF panel, you must take care to:

- Clear the entire long name before pressing ENTER to display a selection list.
- Clear any parts of the old name that extend beyond a shorter name that you type over the original name.

If you only clear the visible part of a long name input field, the rest of the name remains in the ISPF variable, and it will interfere with the processing of subsequence input until you exit the panel. (This is an ISPF behavior, not a ZMF shortcoming.)

TIPS The quickest way to ensure that you have cleared a long name input field is to press PF4 to display the entire long name in a pop-up panel, erase all lines that contain parts of the long name, and press PF3 to return to the original panel.

Alternate Panel - LONG and XLONG

In many ChangeMan ZMF functions that display a list of components, you can invoke an alternate panel that displays the component name on a separate line so there is room to show more characters of a long component name.

For example, if you have zFS library types in a package, then the **STAGE: *package* Components** panel (CMNSTG01) is displayed to show the components in the package.

```
CMNSTG01                  STAGE: ACTP000057 Components          Row 1 to 3 of 3
Command ===> _____________________________________________ Scroll ===> CSR

   Name              + Type Status   Changed         Procname User     Request
__ averylongnamegoesh JVS  ACTIVE   20150315 161850 CMNJAVA  USER015
__ hw001.java         JVS  ACTIVE   20150315 152357 CMNJAVA  USER015
__ org/jdom/xpath/jhf JVS  ACTIVE   20150315 165512 CMNJAVA  USER015
****************************** Bottom of data ******************************
```

This panel CMNSTG01 shows only 18 characters of each component name. The panel permits scrolling left and right via PF10 and PF 11, and Expand (PF4) in the Name field so you can see the rest of the component name.

However, if you type **LONG** on the **Command** line and press **ENTER**, a different **STAGE: *package* Components** panel (CMNSTG14) is displayed, which shows each component zFS path, up to 44 characters, on a line below the rest of the component information.

```
CMNSTG14                  STAGE: ACTP000057 Components         Row 1 to 3 of 3
Command ===> _____________________________________________ Scroll ===> CSR

   Name             + Type Status   Changed         Procname User     Request
                      Org  Input dataset name                + Target lib
__ averylongnamegoesh JVS  ACTIVE   20150315 161850 CMNJAVA  USER015
                      zFS  /cmntp/s6/ACTP/#000057/d/JVS                JVL
__ hw001.java         JVS  ACTIVE   20150315 152357 CMNJAVA  USER015
                      zFS  /cmntp/s6/ACTP/#000057/d/JVS                JVL
__ org/jdom/xpath/jhf JVS  ACTIVE   20150315 165512 CMNJAVA  USER015
                      zFS  /cmntp/s6/ACTP/#000057/d/JVS                JVL
****************************** Bottom of data ********************************
```

This panel still only shows up to 18 characters of a long component name. This panel also offers scrolling and zoom in the Name field if you still cannot see the entire component name.

To return to the original component list panel, type **SHORT** on the **Command** line and press **ENTER**.

Prior to the release of ChangeMan ZMF 7.1 with long names, some panels already responded to the **LONG** command by displaying a panel with additional information. In some of these cases, you can type **XLONG** in the **Command** line and press **ENTER** to display a special panel for long names.

If you type **XLONG** on the **Command** line and press **ENTER**, a further **STAGE: *package* Components** panel (CMNSTG24) is displayed. This panel shows each component name on a line above the component information, with the staging directory path on a third line.

```
CMNSTG24                  STAGE: ACTP000057 Components         Row 1 to 3 of 3
Command ===> _____________________________________________ Scroll ===> CSR

   Name             + Type Status   Changed         Procname User     Request
                      Org  Input dataset name                + Target lib
__ averylongnamegoeshereforjava.java
                      JVS  ACTIVE   20150315 161850 CMNJAVA  USER015
                      zFS  /cmntp/s6/ACTP/#000057/d/JVS                JVL
__ hw001.java
                      JVS  ACTIVE   20150315 152357 CMNJAVA  USER015
                      zFS  /cmntp/s6/ACTP/#000057/d/JVS                JVL
__ org/jdom/xpath/jhfhth40long.java
                      JVS  ACTIVE   20150315 165512 CMNJAVA  USER015
                      zFS  /cmntp/s6/ACTP/#000057/d/JVS                JVL
****************************** Bottom of data ********************************
```

This panel shows up to 75 characters of a long component name. This panel also offers scrolling and zoom in the **Name** field if you still cannot see the entire component name. You can scroll and zoom on the staging directory path.

On this panel you can use the **LONG** command to display the CMNSTG14 panel or **SHORT** to display the CMNSTG01 panel.

TIP If you want to see the component name on a separate line, try **XLONG** first. If short message INVALID SELECTION CODE is displayed, try command **LONG**.

Right Justified Long Names in ERO

ERO uses dynamic ISPF panels to make lists of components for functions like checkin, retrieve, query component, test area, test release, and promotion. ISPF does not support scroll RIGHT and LEFT on dynamic panels.

To display as much meaningful information in the long name column as possible where scroll is not supported, ERO right justifies long names on dynamic panels, truncating any extension on the file name.

In this example, release package JZFS00023 contains the following files with names that include a partial path:

```
/org/jdom/adapters/package.html
/org/jdom/filter/package.html
/org/jdom/input/package.html
/org/jdom/output/package.html
/org/jdom/package.html
/org/jdom/transform/package.html
```

See how these component names are displayed on ERO panel CMNCKI02, right justified with the extension *html* stripped off.

```
CMNCKI02  RELEASE CHECKIN JZFS000023 COMPONENTS ------- Row  000001 Of  000006
COMMAND ===>                                                SCROLL ===> HALF
   COMPONENT NAME   TYPE  STATUS       CHANGED     PROCNAME   ID      AREA ID

 Line Command: S-Select          ( SETALL Select all, SETOFF De-select all )

_ adapters/package  HTH   ACTIVE   20101124 141227          USER240   ACCTPAY
_ m/filter/package  HTH   ACTIVE   20101124 141231          USER240   ACCTPAY
_ om/input/package  HTH   ACTIVE   20101124 141232          USER240   ACCTPAY
_ m/output/package  HTH   ACTIVE   20101124 141234          USER240   ACCTPAY
_ org/jdom/package  HTH   ACTIVE   20101124 141235          USER240   ACCTPAY
_ ransform/package  HTH   ACTIVE   20101124 141237          USER240   ACCTPAY
****************************** Bottom of Data ********************************
```

NOTE Long names on ERO dynamic panels may be displayed using the PF4 zoom function to display the entire name in a pop-up window.

Case Sensitive Fields

By default, all ISPF panel input fields are folded to upper case, regardless of the case you type. However, zFS path names and file names are case sensitive. For example, these are three different files:

- FirstJavaComponent.java
- firstjavacomponent.java
- FIRSTJAVACOMPONENT.java

ChangeMan ZMF uses two methods to control the case sensitivity of input fields on ISPF panels.

- **Data Set Type** in library type definitions
- **Mixed Case** parameter on component list

Data Set Type

The global and application **Library Types Part 2 of 2** panel includes the **Data Set Type** field, with valid values of **LIBRARY** for PDSE, **PDS** or blank for PDS or **zFS**. When you type a component name or directory path for a data set type zFS, case is preserved and stored in ZMF repositories. When component name or directory path are displayed for data set type zFS, the case that is stored in ZMF repositories is displayed unchanged on ISPF panels.

For example, when you type information on the **application/site - Promotion Libraries** panel in application administration, ChangeMan ZMF uses the library type to determine whether the data should default to upper case or be processed exactly as you enter it.

In this example, the library names and directory paths for promotion are all entered in lower case (panel shows one JCL and one zFS libraries - JCL is uppercase).

```
CMNLRPM3             ACTP/SERT6 - Promotion Libraries          Row 2 to 8 of 8
Command ===> _____________________________________________ Scroll ===> CSR

Promotion name: S6P1UT      Level: 10

         Syslib
     Lib exclude Target libraries
____ JCL   Y     CMNTP.S6.V810.PROM.S6P1UT.JCL                  + Shadow
                 CMNTP.S6.V810.PROM.S6P1UT.JCL                  + Library 1
                 ____________________________________________   + Library 2
                 ____________________________________________   + Library 3
____ JVS   N     /cmntp/s6/actp/prom10/jvs                      + Shadow
                 /cmntp/s6/actp/prom10/jvs                      + Library 1
                 ____________________________________________   + Library 2
                 ____________________________________________   + Library 3
...
```

When you press **ENTER**, the data set names for library type JCL are changed to upper case and stored that way on the package master. However, the zFS path names for library type JVS (Java) are left exactly as you entered them, and they are stored in mixed case on the package master.

Mixed Case

On list parameter panels where you specify filter criteria for building a component list, you can control how case is used in component name selection by setting the Mixed Case field.

```
/  Mixed case
```

By default, the text you enter in a component name field is folded to upper case before it is matched against component names. However, if you select the **Mixed Case** field, then the value you enter is compared exactly to the component name in whatever file or directory is being searched.

For example, package ACTP000054 contains these two components:

```
CMNSTG01                  STAGE: ACTP000050 Components         Row 1 to 2 of 2
Command ===>                                                  Scroll ===> CSR

   Name             + Type Status   Changed         Procname User     Request
   acpdoc60           HTH  ACTIVE   20150315 212025          USER015
   ACPDOC60           DOC  ACTIVE   20150315 211855          USER015
****************************** Bottom of data ********************************
```

You can filter the components displayed on the **Stage: *package* Components** panel by first setting selection criteria on the **Component List Parameters** panel (CMNSTG12). If you type the **Component Name** field in lower case on the **Component List Parameters** panel and select the **Mixed case** field, then the filter is case sensitive, and only the HTH component is listed.

Selection criteria:

```
CMNSTG12                   Component List Parameters
Command ===> ____________________________________________________

         Package: ACTP000050      Status: DEV      Install date: 20150318

Component name  . . . . . acpdoc60_____________________________________   +
Component type  . . . . . ___
Language  . . . . . . . . ________
Component status  . . . . . _          (1-active,   2-incomplete,
                                        3-checkout, 4-inactive)
Changed date: from  . . . . ________    (yyyymmdd)
              to  . . . . . ________    (yyyymmdd)
Compile procedure . . . . . ________
User  . . . . . ________
Display mode  . . . . . . . S          (S-short, L-long, X-extra long)

Enter "/" to select option
  / Confirm component delete          / Confirm other requests
  _ Display component user options    / Mixed case
  / Comparison report for edit          Text type . . . . $._____
  _ Ignore recompiled components
```

Result:

```
CMNSTG01                  STAGE: ACTP000050 Components         Row 1 to 1 of 1
Command ===> ____________________________________________________ Scroll ===> CSR

   Name             + Type Status   Changed         Procname User     Request
__ acpdoc60           HTH  ACTIVE   20150315 212025          USER015
****************************** Bottom of data ********************************
```

If you type the **Component Name** field in lower case and do not select the **Mixed Case** field, then the component name you typed is folded to upper case, and only the DOC component is listed.

Selection criteria:

```
CMNSTG12                  Component List Parameters
Command ===> ____________________________________________

         Package: ACTP000050      Status: DEV      Install date: 20150318

Component name  . . . . . . acpdoc60                                    +
Component type  . . . . . . ___
Language  . . . . . . . . . ________
Component status  . . . . . _           (1-active,   2-incomplete,
                                         3-checkout, 4-inactive)
Changed date: from  . . . . ________    (yyyymmdd)
              to  . . . . . ________    (yyyymmdd)
Compile procedure . . . . . ________
User  . . . . . . . . . . . ________
Display mode  . . . . . . . S           (S-short, L-long, X-extra long)

Enter "/" to select option
  / Confirm component delete           / Confirm other requests
  _ Display component user options     / Mixed case
  / Comparison report for edit           Text type . . . . $.______
  _ Ignore recompiled components
```

Result:

```
CMNSTG01                 STAGE: ACTP000050 Components         Row 1 to 1 of 1
Command ===> ____________________________________________ Scroll ===> CSR

   Name             + Type Status   Changed         Procname User     Request
__ ACPDOC60           DOC  ACTIVE   20150315 211855          USER015
****************************** Bottom of data ********************************
```

Component Names That Contain A Path

In USS file systems, a "component name" may include a partial directory path because the actual name of the file is unique only within a hierarchy that includes it.

ChangeMan ZMF automatically handles component names that include a directory path. However, when you stage an zFS component from development, you must indicate whether you want to choose a file name from the specified directory or a path name and file name from that directory.

On the **Stage: From Development** panel, the **Expand** field controls what is displayed on a component selection list. When you set the **Expand** field to **Yes**, all files and paths below the subdirectory you specify in the **DSN** field are displayed on the component selection panel.

Here is the stage panel.

```
CMNSTG02                    Stage from Development
Command ===> ____________________________________________

         Package: ACTP000050      Status: DEV      Install date: 20150318

ISPF Library:
   Project . . . . USER015
   Group . . . . . JAVA ____________________________
   Type  . . . . . SRC
   Member  . . . . ________       (Blank/pattern for list; * for all members)

Other partitioned, sequential or zFS dataset:
   DSN . . . . . . /cmntp/s4/v710/base/jzfs/jav/lvl-0/____________________   +
   Org . . . . . . ___            (PDS, Seq, PAN, LIB, Oth, zFS)

Library type . . . . . JVS        (Blank for list)
Stage name . . . . . . ____________________________________________________   +
Stage mode . . . . . . 1          (1-Online, 2-Batch)

Enter "/" to select option
  / Expand zFS subdirectories
  / Confirm request
  _ Lock component
  _ Display component user options
```

Here is the resulting component selection list.

```
CMNSTG23                      Stage from zFS file             Row 1 to 6 of 6
Command ===>                                                 Scroll ===> CSR

  Input filename
  /cmntp/s4/v710/base/jzfs/jav/lvl-0/                                        +

  Name       +          Function Created     Changed           Size  User
  jhfjav40.java                  2010/07/20  2010/11/05 16:30  00126 SERT
  jhfjav50.java                  2010/11/05  2010/11/05 16:49  00126 SERT
  jhfjav60.java                  2010/11/05  2010/11/05 17:23  00126 SERT
  org/jdom/xpath/jhfj            2010/07/20  2010/07/01 18:53  00151 SERT
  org/jdom/xpath/jhfj            2010/07/20  2010/07/01 18:54  00155 SERT
  org/jdom/xpath/jhfj            2010/07/20  2010/07/01 18:54  00154 SERT
****************************** Bottom of data ******************************
```

Notice that after the first three files, three more components are listed that are in a path of subdirectories below the directory you specified on the **Stage: From Development** panel. If you select one of the files with path names to stage into your package, the component name in the package master and in the component master will include the subdirectories as well as the file name.

Here, **Expand** zFS subdirectories is **not** selected.

```
CMNSTG02                    Stage from Development
Command ===> ____________________________________________

         Package: ACTP000050      Status: DEV      Install date: 20150318

ISPF Library:
   Project . . . . USER015
   Group . . . . . JAVA ____________________________
   Type  . . . . . SRC
   Member  . . . . ________       (Blank/pattern for list; * for all members)

Other partitioned, sequential or zFS dataset:
   DSN . . . . . . /cmntp/s4/v710/base/jzfs/jav/lvl-0/____________________   +
   Org . . . . . . ___            (PDS, Seq, PAN, LIB, Oth, zFS)

Library type . . . . . JVS        (Blank for list)
Stage name . . . . . . ____________________________________________________   +
Stage mode . . . . . . 1          (1-Online, 2-Batch)

Enter "/" to select option
  _ Expand zFS subdirectories
  / Confirm request
  _ Lock component
  _ Display component user options
```

The resulting component selection panel displays only the three components that are contained as files in the directory you specified on the **Stage: From Development** panel.

```
CMNSTG23                      Stage from zFS file             Row 1 to 3 of 3
Command ===>                                                 Scroll ===> CSR

  Input filename
  /cmntp/s4/v710/base/jzfs/jav/lvl-0/                                        +

  Name       +          Function Created     Changed           Size  User
  jhfjav40.java                  2010/07/20  2010/11/05 16:30  00126 SERT
  jhfjav50.java                  2010/11/05  2010/11/05 16:49  00126 SERT
  jhfjav60.java                  2010/11/05  2010/11/05 17:23  00126 SERT
****************************** Bottom of data ********************************
```

NOTE On the Stage: From Development panel (CMNSTG02), you cannot provide a STAGE NAME when the EXPAND field is selected.

Chapter 3

Configuring ZMF for Java/zFS

This chapter provides instructions for setting up ChangeMan ZMF to manage Java components in zFS directories under Unix System Services.

Configure USS for ZMF

Before you can start making global and application administration entries to support Java application components, you must configure Unix System Services and your security system so that the SERNET / ZMF started task can create a secure environment in an zFS file system in USS.

The instructions in this section describe commands for z/OS Security Server RACF. If you use CA ACF2 or CA Top Secret, consult with your security administrator to determine the corresponding actions that are required in those security systems to accomplish the same objectives.

In the commands that follow, these conventions are used:

- **SERUSER** is the user-id assigned to the SERNET / ZMF started task.
- **SERGRP** is the RACF group assigned to the SERNET / ZMF started task.
- **/serdir** is the high-level zFS directory created for user-id SERUSER.

Set ZMF User-ID Security for USS File Systems

Prior to Version 7.1, the ChangeMan ZMF Installation Guide directed you to create an OMVS segment with UID(0) for the userid assigned to the SERNET / ZMF started task. Starting with ZMF 7.1, SERNET and ChangeMan ZMF can perform all necessary USS functions with more restrictive privileges.

Adjust USS privileges for SERUSER, the user-id (owner) of SERNET / ZMF started tasks:

1 Assign a non-zero UID to SERUSER by manually assigning the next available value:

```
ALTERUSER SERUSER OMVS(UID(xxx))
```

2 Permit access for SERUSER to two resources so it can manage zFS in USS:

```
PERMIT BPX.SERVER CLASS(FACILITY) ID(SERUSER) ACCESS(UPDATE)
PERMIT SUPERUSER.FILESYS CLASS(UNIXPRIV) ID(SERUSER) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

3 Ensure that the SERUSER default group SERGRP has a GID:

```
ALTERGROUP SERGRP OMVS(GID(YYY))
```

NOTES In early editions of this manual, instructions for setting up security for USS file systems included steps to activate profile FILE.GROUPOWNER.SETGID in the UNIXPRIV class to ensure that anything created by the started task user ID inherits the started task GID. The intention was to guarantee that the entire zFS directory/file structure has consistent owner/group assignments.

It has been determined that the default behavior for USS is to propagate group ownership downwards for all directories and files that ChangeMan ZMF creates under the top level ZMF directory. Therefore, profile FILE.GROUPOWNER.SETGID is not required, and instructions for enabling it are removed from later editions of this book.

Create Top-Level Directory for ZMF zFS Files

Create a directory where ZMF can create and manage zFS directories and files. This is not necessarily the directory where baseline, promotion, and production libraries reside, but staging libraries should be created in this directory.

1 Create a top-level directory for USS files that will be managed by ChangeMan ZMF instances that are owned by user-id SERUSER:

   a Create a top level directory:

```
/serdir
```

   b Change the owner and group owner of /serdir to SERUSER and SERGRP respectively, and propagate the change to all directories and files that may have been created under the top-level directory.

```
chown -R sert:cmntp /serdir
```

   c Set the permission bits for /serdir to allow SERUSER and ZMF Administrators in group SERGRP to READ, WRITE and EXECUTE, and allow all others to READ EXECUTE.

```
chmod 755 /serdir
```

2 Create and mount the zFS data set for /serdir:

   a Create the zFS data set for /serdir.

   b Mount the zFS data set at /serdir:

```
MOUNT FILESYSTEM('zFS datasetname') mountpoint('/serdir') TYPE(ZFS) MODE(RDWR)
     AUTOMOV
```

   c Add the mount of /serdir to the BPXPRMnn member in the primary SYS1.PARMLIB to ensure that the mount for /serdir is executed at IPL.

Adjust Security on zFS Files Managed by ZMF

For any path entered into ZMF Administration for baseline, production, or promotion libraries, adjust the permissions to give the ZMF started task ownership with permission 755.

NOTES Exit exit program CMNEX093 makes it possible to override the ChangeMan ZMF default 755 permission for USS staging libraries. See the comments in the CMNEX093 program source delivered in the ASMSRC library.

Increase MAXPROCUSER or PROCUSERMAX

Unix System Services limits the number of simultaneous processes running under a userid (UID), even if the processes are initiated from multiple address spaces. ChangeMan ZMF can potentially create a large number of parallel processes inside USS as Java build jobs are file tailored and submitted under the ZMF UID.

The number of parallel processes per UID is limited by the MAXPROCUSER entry in BPXPRMxx. When this value is exceeded, USS fails the creation of new processes, which will cause jobs or tasks inside ZMF to fail. When creation of a process is failed, message BPXP005I is issued with a return code of 70 (unless suppressed by USS specifications).

A rough calculation for the number of processes required is:

```
   2 * (# of parallel USS related jobs *executing* concurrently)
+  2 * (# of concurrent ZMF users logged on)
```

If you do not want to increase the global limit using the MAXPROCUSER entry in BPXPRMxx, use the PROCUSERMAX parameter on the OMVS segment of the RACF profile for the ChangeMan ZMF userid.

```
ALTUSER userid OMVS(PROCUSERMAX(nnnn))
```

This command is described in "Unix System Services Planning", Section 3.

Enable CMNEX026 for LSH Listing

Listings from Java build processing are stored in zFS files in library type LSH. Library type LSH is coded in Java build skeletons, but it is not a reserved library type.

The following code is delivered in CMNEX026 to allocate an LSH staging library when language JAVA is processed.

```
*
* The following sample ensures that the zfs based listings libtype
* is allocated whenever a language name of JAVA is used
*
         CLC   X26$LANG,=CL8'JAVA' java language?
         JNE   X26$O010            .no,  skip java
         MVI   X26$WORK,C'Y'       indicate something added
         MVC   0(3,R2),=CL3'LSH'   alloc lsh for java
         LA    R2,3(,R2)           bump for next entry
```

Follow the instructions at the top of exit program CMNEX026 to enable the exit. There is no need for you to add Java library types to table X26@LTYP. You may also like to examine CMNEX032.

Configure ZMF Global and Application Administration

This section shows you entries in ChangeMan ZMF global and application administration that are required to manage Java components and work in zFS.

When you enter zFS paths and directories for staging and baseline files, those paths should be under the exclusive control of the user-id assigned to the SERNET / ZMF started task. We recommend that the path be under the high-level directory you defined in "Create Top-Level Directory for ZMF zFS Files" on page 29.

The paths for production and promotion directories should also be under the exclusive control of the SERNET / ZMF started task, although your environment may require some sharing of write permission.

Global Staging Library Model zFS Name

When you check out a component that is stored in an zFS file, the staging library allocated for that component must be an zFS directory. On the **Global Parameters – Part 7 of 8**

panel (CMNGGP07), enter a Dev model zFS name to tell ChangeMan ZMF how to structure the staging path and directory name.

```
CMNGGP07                Global Parameters - Part 7 of 8
Command ===> ____________________________________________

Staging library model dataset names
 Dev model dsname  . . . CMNTP.S6.????.STG6.#######
 Dev model zfs name  . . /cmntp/s6/????/#######/d                +

   "????" is placeholder for application name.
   "#######" is placeholder for package number (.#000123).

 zfs temp folder . . . . /u/sert/C001/tmp/s6                     +

 Package master  . . . . CMNTP.S6.V810T06.CMNZMF.CMNPMAST
 Cpnt mstr (short) . . . CMNTP.S6.V810T06.CMNZMF.CMNCMPNT
 Cpnt mstr (long)  . . . CMNTP.S6.V810T06.CMNZMF.CMNCMPNL
 Delay file  . . . . . . CMNTP.S6.V810T06.CMNZMF.CMNDELAY
 Ser#parm  . . . . . . . CMNTP.SER810.C6.TCPIPORT
 Impact analysis . . . . CMNTP.S6.V810T06.CMNZMF.IADSP

 Email server  . . . . . mail.serena.com                  +
         port  . . . . . 00025
```

TIP You can use the same general naming pattern you use for PDS(E) libraries, substituting a slash for each period. However, the path should start with the high-level directory you defined in the Security for zFS section of this document.

NOTE ChangeMan ZMF does not use the information in the **Prd model zfs name** field on this panel. Instead, the **Prd staging model zfs** on the ***site*** **Site Information - Part 2 of 2** panel (where ***site*** is a variable value) is used.

Global zFS Temporary Folder

Some component processes in ZMF create temporary files. When you are working with a package component that is stored in an zFS file, temporary files will also be zFS files.

You specify a directory path for temporary zFS files on the **Global Parameters – Part 7 of 8** panel (CMNGGP07) shown above in field **zfs temp folder**. This field initially shows a top-level directory value of /tmp. Add at least one subdirectory to create a path where ZMF will create zFS temporary files.

```
zfs temp folder    ===> /tmp/s4
```

Global Site zFS Production Staging Model DSNAME

The ***site*** **Site Information – Part 2 of 2** panel (CMNGRST2) requires the **PRD staging model zFS** path for a remote site.

This example of the ***site*** **Site Information – Part 2 of 2** panel (CMNGRST2) shows the **Prd staging model zFS** path for a development site, where the path is the same as the **Dev model zFS name** on the **Global Parameters - Part 7 of 8** panel (CMNGGP07).

```
CMNGRST2            SERT6 Site Information - Part 2 of 2
Command ===>

ChangeMan ZMF subsytem id  . . 6
Logical unit name  . . . . . . BUCKS                                    +
JES node name  . . . . . . . . C001
Default unit name  . . . . . . SYSDA    (Generic disk unit)
Default volume serial  . . . .
ChangeMan ZMF delay file . . . CMNTP.S6.V810T06.CMNZMF.CMNDELAY
Ser#parm . . . . . . . . . . . CMNTP.SER810.C6.TCPIPORT
Prd staging model dsname . . . CMNTP.S6.????.STG6.#######
Prd staging model zFS  . . . . /cmntp/s6/????/#######/d                 +
Transmission vehicle . . . . . IEBCOPY  (IEBCOPY or Other)
Time difference  . . . . . . . +0000    (+/- HHMM)
IP address or DNS name . . . .                                          +
Port . . . . . . . . . . . . .

Site job statement information:
   //CMNSTART JOB ,'SERT6 SITE',
   //             CLASS=A,MSGCLASS=X
   //*
   //* SITE SERT6 JOB STATEMENTS
```

This example shows the **Prd staging model zFS** path for a remote site, where the path is different from the **Dev model zFS name** on the **Global Parameters - Part 7 of 8** panel (CMNGGP07).

```
CMNGRST2            SERT6P1 Site Information - Part 2 of 2
Command ===>

ChangeMan ZMF subsytem id  . . 7
Logical unit name  . . . . . . BUCKS                                    +
JES node name  . . . . . . . . C001
Default unit name  . . . . . . SYSDA    (Generic disk unit)
Default volume serial  . . . .
ChangeMan ZMF delay file . . . CMNTP.S7.V810.CMNZMF.CMNDELAY
Ser#parm . . . . . . . . . . . CMNTP.SER810.C6.TCPIPORT
Prd staging model dsname . . . CMNTP.S7.????.STG7P1.#######
Prd staging model zFS  . . . . /cmntp/s6/????/#######/p1                +
Transmission vehicle . . . . . IEBCOPY  (IEBCOPY or Other)
Time difference  . . . . . . . +0000    (+/- HHMM)
IP address or DNS name . . . .                                          +
Port . . . . . . . . . . . . .

Site job statement information:
   //CMNSTART JOB ,'SERT6P1 SITE',
   //             CLASS=A,MSGCLASS=X
   //*
   //* SITE SERT6P1 JOB STATEMENTS
```

Global and Application Library Types

Define like-source, like-load, and like-PDS library types for your Java components in global and application library types. This ***application* - Library Types Part 1 of 2** panel (CMNCLLT0) from a Java-only application shows the relationship between like-source and like-load Java library types.

```
CMNCLLT0              JZFS - Library Types Part 1 of 2       Row 1 to 11 of 11
Command ===> _____________________________________________ Scroll ===> CSR

     Lib                                          Order Lke Seq Defer Target Sel
     type Description                        +                        type   Opt
____ HTH  zFS resident HTML                   0     P   ___   Y     ___    __
____ JAR  Java Archives                       0     L   ___   Y     ___    __
____ JCF  Java Class files                    0     L   ___   Y     ___    __
____ JCL  Execution JCL                       0     P   ___   Y     ___    __
____ JCT  Java JAR Build Control              0     S   ___   Y     JAR    __
____ JVL  zFS - JAVA executable class         0     L   ___   Y     ___    __
____ JVS  zFS - JAVA source type              0     S   ___   Y     JVL    __
____ JVT  zFS - text type                     0     P   ___   Y     ___    __
____ LSH  zFS Listings                        0     P   ___   Y     ___    __
____ WAR  Java Web Archives                   0     L   ___   Y     ___    __
____ WCT  Java WAR Build Control              0     S   ___   Y     WAR    __
****************************** Bottom of data ********************************
```

IMPORTANT! In some cases, library type definitions for Java components may not be intuitive because a component must be defined as like-source to initiate a build process, and the output of a build process must be defined as like-load so a source-to-load relationship can be recorded. How the library types in this example work in Java build processing is explained in Chapter 4, "Working With Java" on page 39.

The global and application **Library Types Part 2 of 2** panels (CMNCLLT1) generally look like this with only **Data Set Type** specified in **Staging Dataset Attributes**.

```
CMNCLLT1             JZFS - Library Types Part 2 of 2
Command ===> ____________________________________________

Library type: HTH -  zFS resident HTML
Like value:   P     Defer value:   Y

Staging dataset attributes:
 Generic unit name . . . . . ________ (Generic group name or unit)
 Volume serial . . . . . . . ______   (Required if generic unspecified)
 Space units . . . . . . . . ___      (trk, cyl or blk)
 Primary quantity  . . . . . ________ (In above units)
 Secondary quantity  . . . . ________ (In above units)
 Directory blocks  . . . . . _____
 Record format . . . . . . . ___
 Record length . . . . . . . _____
 Block size  . . . . . . . . _____
 Data set type . . . . . . . ZFS___   (library, pds, zfs or blank)
 Extended attributes . . . . ___      (no, opt or blank)
 Save staging versions . . . . ALWAYS (always/none/prompt)

Enter "/" to select option:
  _ Checkout component description
  _ Checkout component activity file   Library type . . . . ___
```

NOTE Staging Versions may be enabled for any text component, which includes zFS library types HTH, JCT, JAV, and WCT in the example provided here.

Global and Application Language and Compile Procedures

These are the only two language/procedure combinations required for Java.

```
CMNCLPRC                 JZFS - Compile Procedures             Row 1 to 2 of 2
Command ===> _____________________________________________ Scroll ===> CSR

     Language  Procedure  Description                                   Order
____ JAVA      CMNJAR     Create Java archive                           0
____ JAVA      CMNJAVA    Stage Java source                             0
****************************** Bottom of data ********************************
```

Define the JAVA language and the procedures shown here on the global and application **Language Names** and **Compile Procedures** panels.

Application Baseline Libraries

On the ***application*** **- Baseline Configuration Part 1 of 2** panel (CMNCBAS1), specify a **Baseline Storage Means** of **H** for all Java baseline repositories. Prior versions of baselined Java components are full copies, not delta decks.

```
CMNCBAS1     JZFS -  Baseline Configuration Part 1 of 2      Row 1 to 10 of 10
Command ===> _____________________________________________ Scroll ===> CSR

                                 Baseline
                      Install    storage
     Type  Levels  in prod    means
____ HTH     3         N       H
____ JAR     3         Y       H
____ JCF     3         N       H
____ JCT     3         N       H
____ JVL     2         N       H
____ JVS     2         N       H
____ JVT     2         N       H
____ LSH     3         Y       H
____ WAR     3         Y       H
____ WCT     3         N       H
****************************** Bottom of data ********************************
```

In the example shown here, only executable library types JAR and WAR are installed in production execution libraries (directories), along with build listings in library type LSH to provide a diagnostic reference in case of a production problem.

On the ***application*** **- Baseline Configuration Part 1 of 2** panel (CMNCBAS1), type line command **S** by a library type to display the ***application*** **- Baseline Configuration Part**

2 of 2 panel (CMNCBAS2) where you allocate new baseline paths and directories or verify existing directories.

```
CMNCBAS2    JZFS -  Baseline Configuration Part 2 of 2         Row 1 to 2 of 2
Command ===> _____________________________________________ Scroll ===> CSR

   Library type:        JVS
   Levels maintained:   2
   Storage means:       zFS

   Lvl Dataset name      +                                Status
_ -000 /cmntp/s6/jzfs/base/jvs/lvl-0
_ -001 /cmntp/s6/jzfs/base/jvs/lvl-1
****************************** Bottom of data ********************************
```

CAUTION! JAR and WAR build processing uses path names to collect files to be included. Exercise care when defining directory paths for baseline and promotion libraries. The following naming structure is common for MVS baseline libraries, but it would result in -1 and -2 level libraries being included in JAR or WAR builds.

```
-000 /cmntp/s4/v710/base/jzfs/jav
-001 /cmntp/s4/v710/base/jzfs/jav/lvl-1
-002 /cmntp/s4/v710/base/jzfs/jav/lvl-2
```

On the ***application* - Baseline Configuration Part 1 of 2** panel (CMNCBAS2), type line command **A** next to a path name to create the path.

```
CMNCBAS2    JZFS -  Baseline Configuration Part 2 of 2         Row 1 to 2 of 2
Command ===> _____________________________________________ Scroll ===> CSR

   Library type:        JVS
   Levels maintained:   2
   Storage means:       zFS

   Lvl Dataset name      +                                Status
_ -000 /cmntp/s6/jzfs/base/jvs/lvl-0                     *Allocated
A -001 /cmntp/s6/jzfs/base/jvs/lcl-1
****************************** Bottom of data ********************************
```

NOTE When you allocate an zFS baseline path and directory from the ***application* – Baseline Configuration Part 2 of 2** panel, there is no additional panel for library attributes as there is for PDS(E) library allocations.

On the ***application* - Baseline Configuration Part 1 of 2** panel (CMNCBAS2), type line command **V** next to a path name to verify the path and display the **zFS Information** panel if the path exists.

```
CMNCBAS2    JZFS -  Baseline Configuration Part 2 of 2         Row 1 to 2 of 2
Command ===>                                               Scroll ===> CSR

   Library type:        JVS
   Levels maintained:   2
   Storage means:       zFS

   Lvl Dataset name      +                                Status
  -000 /cmntp/s6/jzfs/base/jvs/lvl-0                     *Verified
V -001 /cmntp/s6/jzfs/base/jvs/lcl-1
****************************** Bottom of data ********************************
```

```
CMNDAIH                          ZFS Information
Command ===> ____________________________________________________________

Pathname . . . /cmntp/s6/jzfs/base/jvs/lvl-2                              +

Permissions  . 755
File size  . . 00008192
File owner . . SERT
Group owner  . CMNTP
Created  . . . 2015-03-22   21:25:49
Modified . . . 2015-03-22   21:25:49
Accessed . . . 2015-03-22   21:25:49
Link count . . 00002
```

Application Production Libraries

On the ***application - site*** **Production Libraries** panel (CMNCPRDL), specify paths for zFS components that you want copied to production execution libraries at package installation.

```
CMNCPRDL            JZFS - SERT6 Production Libraries          Row 5 to 7 of 7
Command ===> ____________________________________________ Scroll ===> CSR

     Type Production dataset name    +
          Temporary dataset name     +
          Backup dataset name        +
____ JVS  /cmntp/s6/jzfs/prod/jvs
          /cmntp/s6/nullfile
          /cmntp/s6/jzfs/prod/jvs/backup
____ JVT  /cmntp/s6/jzfs/prod/jvt
          /cmntp/s6/nullfile
          /cmntp/s6/jzfs/prod/jvt/backup
____ JVL  /cmntp/s6/jzfs/prod/jvl
          /cmntp/s6/nullfile
          /cmntp/s6/jzfs/prod/jvl/backup
****************************** Bottom of data ********************************
```

IMPORTANT! zFS components cannot be included in a temporary package. The **Temporary Dataset Name** is just a placeholder to satisfy ISPF panel edits.

Application Promotion Libraries

On the ***application/site* – Promotion Libraries** panel (CMNLRPM3), specify paths for test libraries you want to populate from staging directories.

```
CMNLRPM3            JZFS/SERT6 - Promotion Libraries          Row 3 to 8 of 8
Command ===> _____________________________________________ Scroll ===> CSR

Promotion name: S6P1UT      Level: 10

         Syslib
     Lib exclude Target libraries
____ JVS   N     /cmntp/s6/jzfs/prom10/jvs                    + Shadow
                 /cmntp/s6/jzfs/prom10/jvs                    + Library 1
                 ____________________________________________ + Library 2
                 ____________________________________________ + Library 3
     JVT   N     /cmntp/s6/jzfs/prom10/jvt                    + Shadow
                 /cmntp/s6/jzfs/prom10/jvt                    + Library 1
                 ____________________________________________ + Library 2
                 ____________________________________________ + Library 3
     JVL   N     /cmntp/s6/jzfs/prom10/jvl                    + Shadow
                 /cmntp/s6/jzfs/prom10/jvl                    + Library 1
                 ____________________________________________ + Library 2
                 ____________________________________________ + Library 3
...
```

Deploy Java Applications To WebSphere

ChangeMan ZMF can build a Java Web Application (EAR file) and deploy it to WebSphere on a z/OS system. Deployment means that the application is delivered to a location accessible to WebSphere, and a post-processing task uses published WebSphere APIs to refresh the running application.

ISPF file tailoring of ZMF skeleton CMNJAR generates batch job JCL to create JAR/WAR/EAR files. Standard ZMF promotion, install, and backout skeletons deliver EAR files to a WebSphere accessible location (typically the webapps folder). Three skeletons create JCL for post-processing of delivered EAR files:

- CMN$$WSP imbedded in promotion skeleton CMN$$PRO
- CMN$$WSI imbedded in install skeleton CMN20
- CMN$$WSB imbedded in backout skeleton CMN50

CMN$$WSP, CMN$$WSI and CMN$$WSB call WebSphere scripting APIs to refresh the application being updated. These skeletons must be customized for each environment to set variables for each application/libtype. The following table lists the variables that must be customized in each of these skeletons. Most of these values can be obtained from your WebSphere Application Server administrator.

Variable	Description
WSHOST	WebSphere HOST System - The host where WebSphere is located
WSPORT	WebSphere SOAP port - The WebSphere SOAP port
WSUSER	WebSphere Administrative ID
WSPASS	WebSphere Administrative id password

Variable	Description
WSSERVER	WebSphere Server - The WebSphere server name where the application is located
WSNODE	Websphere Node - The WebSphere Node name where the application is located
WSAPPL	WebSphere Application - The WebSphere application name
WSSCRIPT	Jython Script file to execute - The script file to execute. The default is zmfws.jy. The path to this file also must be updated.

Also, the following details must be configured in each skeleton:

- Configure path to wsadmin - The path to the WebSphere wsadmin tool.
- Add component in CMN$$WSI - Component names must be added for the install skeleton.

Chapter 4

Working With Java

Packages containing Java components in zFS files must follow the standard ChangeMan ZMF package life cycle. However, some of the processes in that life cycle are different for Java components.

Java Build Processing

ChangeMan ZMF build processing requires a like-source library type for input and a like-load library type for output. When you are working with Java applications in ZMF, the like-source component may not actually be program source code, and the like-load output is an executable that is not a load module.

All files, including build listings, are zFS files.

Compile Java Source

This build process transforms Java source into a class file.

Components

Input / Output	Description	Like-Type	Extension	Library Type Example
Input	Java source	S	.java	JAV
Output	Java class file	L	.class	JCF
Output	zFS build listing	P	.list	LSH

Process

- Language: JAVA
- Procedure: CMNJAVA

Stage and recompile panels allow mixed case data in the COMPILE PARMS field when the LANGUAGE field is JAVA. Skeletons pass COMPILE PARMS options to the javac compiler, and COMPILE PARMS options are stored in component history for use in subsequent builds for the component.

CAUTION! The ChangeMan ZMF Java build process requires option -verbose, which is hard coded in build skeletons (delivered CMN$$JVA). If this option is nullified by an entry in the COMPILER OPTION field, the component build will fail.

Build Java Archive

This build process follows instructions in a file of JAR control statements to create a JAR file.

Components

Input / Output	Description	Like-Type	Extension	Library Type Example
Input	JAR control statements	S		JCT
Output	Java archive (JAR)	L	.jar	JAR
Output	zFS build listing	P	.list	LSH

Process

- Language: JAVA
- Procedure: CMNJAR

JAR Control Statements

This table describes JAR control statements.

Keyword	Description
*	Comment Syntax: * in position 1 Comment records are read and printed in the CMNPRINT data set, but content is ignored by JAR build processing.
jarname	Specifies the file name of the output JAR file that is written to the package staging library for the target JAR library type. Syntax: jarname=*filename* *filename* The output JAR name with extension. The JAR name may include a fragment of a path name. The *filename* string cannot exceed the maximum for the file system in use (255 characters in the current file systems).
jarpath	Specifies a ZMF library type to be included in the JAR file. Files are copied from application baseline libraries and package staging libraries for the library type. Files in subdirectories are copied. Syntax: jarpath=*libtyp*,*qualifier* *libtype* Three character library type. (Required, upper case.) *qualifier* Fragment of a path name that acts as a filter to limit the scope of the copy to a subset of the files in the library type. (Optional, case sensitive.)

Keyword	Description
jardir	Specifies a ZMF library type to be included in the JAR file. Files are copied from application baseline libraries and package staging libraries for the library type. Files in subdirectories are NOT copied. Syntax: jardir =*libtyp*,*qualifier* *(*Note the space before =.*)* *libtype* Three character library type. (Required, upper case.) *qualfier* Fragment of a path name that acts as a filter to limit the scope of the copy to a subset of the files in the library type. (Optional, case sensitive.)
jarfile	Specifies a ZMF library type and the name of a file to be included in the JAR file. The file is copied from application baseline libraries and package staging libraries for the library type. Syntax: jarfile=*libtype*,*filename* *libtype* Three character library type. (Required, upper case.) *filename* File name with extension, including full path name after baseline or staging library name. (Required, case sensitive.) If *filename* is omitted, no file is copied and no error is reported.

JAR Control Examples

In the examples that follow, these are the contents of application baseline and package staging libraries for library type HTH.

Baseline library: /hthbaseline/

```
/hthbaseline/dir_1/adapter/file_x.html
/hthbaseline/dir_1/adapter/dir_2/file_x.html
/hthbaseline/dir_1/filter/file_x.html
/hthbaseline/dir_1/input/file_x.html
```

Staging library: /stagingmodel/HTH

```
/stagingmodel/HTH/dir_1/adapter/file_x.html
/stagingmodel/HTH/dir_1/adapter/dir_2/file_x.html
```

Example 1:

```
jarpath=HTH
```

Included files:

```
/stagingmodel/HTH/dir_1/adapter/file_x.html
/stagingmodel/HTH/dir_1/adapter/dir_2/file_x.html
/hthbaseline/dir_1/filter/file_x.html
/hthbaseline/dir_1/input/file_x.html
```

Example 2:

```
jarpath=HTH,dir_1/adapter
```

Included files:

```
/stagingmodel/HTH/dir_1/adapter/file_x.html
/stagingmodel/HTH/dir_1/adapter/dir_2/file_x.html
```

Example 3:

```
jardir =HTH
```

Included files:

```
/hthbaseline/dir_1/adapter/file_x.html
/hthbaseline/dir_1/adapter/dir_2/file_x.html
/hthbaseline/dir_1/filter/file_x.html
/hthbaseline/dir_1/input/file_x.html
```

Example 4:

```
jardir =HTH,dir_1/adapter
```

Included files:

```
/stagingmodel/HTH/dir_1/adapter/file_x.html
```

NOTE When a qualifier is specified for keyword jardir, files in subdirectories are excluded.

Build Web Archive

This build process follows instructions in a file of WAR control statements to create a WAR file.

Components

Input / Output	Description	Like-Type	Extension	Library Type Example
Input	WAR control statements	S		WCT
Output	Java Web archive (WAR)	L	.war	WAR
Output	zFS build listing	P	.list	LSH

Process

- Language: JAVA
- Procedure: CMNJAR

WAR Control Statements

The control statements used to build a WAR file are the same as those used to build a JAR file. See "JAR Control Statements" on page 41.

Impact Analysis for Java

The only Impact Analysis relationships recorded for Java are for components included in JAR files, like static subroutines in composite executables.

Use the SUBROUTINE relationship on the **Impact Analysis Of Subordinate Components** panel and the **Component Bill Of Materials** panel.

Package Audit for Java

Package audit detects source-to-load inconsistencies and out-of-date issues in Java components like it does for non-Java components. There is no source-to-copy relationship in Java, and the only subroutine-to-composite relationship is between a Java archive (JAR) and its elements. Unlike MVS subroutines in composite loads, JAR elements may be any like-type.

Long names for Java components require a different package audit report format than is used for non-Java components. However, the conditions checked are similar to the conditions checked for PDS components, and the report contents are similar.

Package Out-of-Sync Conditions for Java

The following table lists the out-of-sync conditions that are analyzed for Java components. For more information, see Chapter 11 "Auditing a Package" in the *ChangeMan ZMF User's Guide*.

Out-of-Sync Condition	Description	RC
SYNCH0!	Package Master contains no record of this component.	0
SYNCH1!	Audit is unable to extract component statistics from the file system to perform one or more of the other SYNCH checks.	0
SYNCH2!	Like-source component was compiled without using the designated compile procedure and options.	12
SYNCH6!	Component in a staging library has no corresponding Component Activity File member in the package.	12
SYNCH7!	Element in a Java archive in a staging library was changed more recently than the Java archive in a **staging** library in the same package.	12
SYNCH8!	Element in a Java archive in a staging library was changed more recently than the Java archive in the **baseline** library.	8
SYNCH9!	Like-load component in a staging library does not match the corresponding like-source component in the package.	12
SYNCH10!	Component in the baseline library has changed since the component was checked out to a package staging library.	12
SYNCH11!	Component in a staging library was changed without using ChangeMan ZMF.	12
SYNCH12!	Component was copied into a staging library without using ChangeMan ZMF, or it is left over from a stage job that abended.	12

Out-of-Sync Condition	Description	RC
SYNCH14!	Package component is not in Active status.	12
SYNCH19!	A package master record points to a package component that is not in the staging library.	12
SYNCH20!	Element in a Java archive in a staging library has a SETSSI that does not match the SETSSI of the version of the element that audit expects would be obtained by the Java archive build from your package staging libraries (or eligible participating package) or from baselines (first found location).	8

Package Audit Report for Java Components

The package audit report for Java components has divisions similar to the package audit report for MVS components:

- Stand alone errors reported against package contents
- Relationship errors between header files and source
- Relationship errors between subroutines and composite executables
- Relationship errors between archives and included components

Each division is divided into sections relating to a common BUN. Heading information is similar to that provided in the non-zFS audit report.

The sample package audit report below validates package ACTP000062 that contains the following components.

```
CMNSTG01                  STAGE: ACTP000062 Components         Row 1 to 4 of 4
Command ===>                                                  Scroll ===> CSR

   Name             + Type Status   Changed          Procname User     Request
   java040.java       JVS  ACTIVE   20150324 190145 CMNJAVA  USER015
   java050.java       JVS  ACTIVE   20150324 190230 CMNJAVA  USER015
   java060.java       JVS  ACTIVE   20150324 190306 CMNJAVA  USER015
   java070.java       JVS  ACTIVE   20150324 190345 CMNJAVA  USER015
****************************** Bottom of data ********************************
```

This is how each component was processed to produce the out-of-sync conditions shown in the sample audit report:

Component	Processing
java40.java	Checkout, edit, and stage.
java50.java	Checkout and stage without changes.
java60.java	Checkout, stage without changes, then baseline from another package.
java70.java	Stage from development.

This is the audit report for the package.

```
ChangeMan(R) ZMF  (8.1.0 - 20141010)  Audit  TUESDAY MARCH 24, 2015    (2015/083) 20:44:26                    Page 1
           ********************************************************************************************
           *Simple Change Package     ===> ACTP000062 Created 2015/03/24 at 18:53:15 by USER015          *
           *Package Installation Date ===> 2015/06/30         Package Status: DEV                        *
           *Component Analysis Type   ===> zFS component list including non-relational errors.          *
           *Library Appl:Libtype      ===> ACTP:LSH ACTR:LSH                                             *
           ********************************************************************************************
           Component Name                     Error      From    Timestamp             Package      User
           _________________________________  _________  _____   ___________________   __________   _______
           java040.JVS.list                              Stage   2015/03/24 19:01:47   ACTP000062   USER015
                                                         Base                          ACTP000061   USER015

           java050.JVS.list                              Stage   2015/03/24 19:02:32   ACTP000062   USER015
                                                         Base                          ACTP000061   USER015

           java060.JVS.list                              Stage   2015/03/24 19:03:07   ACTP000062   USER015
                                                         Base                          ACTP000063   USER015

           java070.JVS.list                              Stage   2015/03/24 19:03:47   ACTP000062   USER015
                                                         Base                          ACTP000061   USER015



ChangeMan(R) ZMF  (8.1.0 - 20141010)  Audit  TUESDAY MARCH 24, 2015    (2015/083) 20:44:26                    Page 2
           ********************************************************************************************
           *Simple Change Package     ===> ACTP000062 Created 2015/03/24 at 18:53:15 by USER015          *
           *Package Installation Date ===> 2015/06/30         Package Status: DEV                        *
           *Component Analysis Type   ===> zFS component list including non-relational errors.          *
           *Library Appl:Libtype      ===> ACTP:JVL ACTR:JVL                                             *
           ********************************************************************************************
           Component Name                     Error      From    Timestamp             Package      User
           _________________________________  _________  _____   ___________________   __________   _______
           java040.class                                 Stage   2015/03/24 19:01:44   ACTP000062   USER015
                                                         Base                          ACTP000061   USER015

           java050.class                                 Stage   2015/03/24 19:02:29   ACTP000062   USER015
                                                         Base                          ACTP000061   USER015

           java060.class                                 Stage   2015/03/24 19:03:04   ACTP000062   USER015
                                                         Base                          ACTP000063   USER015

           java070.class                                 Stage   2015/03/24 19:03:44   ACTP000062   USER015
                                                         Base                          ACTP000061   USER015



ChangeMan(R) ZMF  (8.1.0 - 20141010)  Audit  TUESDAY MARCH 24, 2015    (2015/083) 20:44:26                    Page 3
           ********************************************************************************************
           *Simple Change Package     ===> ACTP000062 Created 2015/03/24 at 18:53:15 by USER015          *
           *Package Installation Date ===> 2015/06/30         Package Status: DEV                        *
           *Component Analysis Type   ===> zFS component list including non-relational errors.          *
           *Library Appl:Libtype      ===> ACTP:JVS ACTR:JVS                                             *
           ********************************************************************************************
           Component Name                     Error      From    Timestamp             Package      User
           _________________________________  _________  _____   ___________________   __________   _______
           java040.java                                  Stage   2015/03/24 19:00:15   ACTP000062   USER015
                                                         Base    2015/03/24 18:07:24   ACTP000061   USER015

           java050.java                       DUPLIC!    Stage   2015/03/24 18:58:11   ACTP000062   USER015
                                                         Base    2015/03/24 18:07:24   ACTP000061   USER015

           java060.java                       SYNCH10!   Stage   2015/03/24 18:58:15   ACTP000062   USER015
                                                         Base    2015/03/24 20:43:06   ACTP000063   USER015

           java070.java                                  Stage   2015/03/24 19:02:25   ACTP000062   USER015
                                                         Base    2015/03/24 18:07:24   ACTP000061   USER015
```

```
      ChangeMan(R) ZMF  (8.1.0 - 20141010)  Audit  TUESDAY MARCH 24, 2015    (2015/083) 20:44:26                  Page 4

                                              Legend and Summary Report
The local level of audit chosen at this point;      4
    4 - Audit is required and the return code must not exceed 4 which implies
        that there are no "out-of-synch" situations within the staging libraries
        nor the baseline libraries but at least one module of a staging library
        is a "duplicate" of its baseline counterpart
Out-of-synch messages (hint - search for "!" marks)
    DUPLIC! (Staging duplicates baseline)      ===> 1
    SYNCH0! (Not in scope of audit or unknown) ===> 0
    SYNCH1! (Cmpnt statistics not available)   ===> 0
    SYNCH2! (Compile/designated proc differ)   ===> 0
    SYNCH3! (Unparsable load module)           ===> 0
    SYNCH4! (cpy/hdr staging problem)          ===> 0
    SYNCH5! (cpy/hdr baseline problem)         ===> 0
    SYNCH6! (Activity file not checked out)    ===> 0
    SYNCH7! (Static subcomponent stage problem) ==> 0
    SYNCH8! (Static subcomponent base problem) ===> 0
    SYNCH9! (Source and load discrepancy)      ===> 0
    SYNCH10! (Version regression problem)      ===> 1
    SYNCH11! (Component hash discrepancy)      ===> 0
    SYNCH12! (Orphan module in staging)        ===> 0
    SYNCH13! (Baseline/staging discrepancy)    ===> 0
    SYNCH14! (Components not in active status) ===> 0
    SYNCH15! (Source to relationship problem)  ===> 0
    SYNCH16! (CPY low-date problem in baseline)===> 0
    SYNCH17! (CPY deleted problem in staging)  ===> 0
    SYNCH18! (LOD deleted problem in staging)  ===> 0
    SYNCH19! (Missing module in staging)       ===> 0
    SYNCH20! (Inconsistent subroutine)         ===> 0
    SYNCH21! (Scr/rename pkg component)        ===> 0
    SYNCH22! (Scratch subcmpnt is in use)      ===> 0
    SYNCH23! (Rename subcmpnt is in use)       ===> 0

Highest return code encountered                ===> 12
CMN3060A - This package has failed the audit.
CMN2696I - PACKAGE ACTP000062 FAILED THE AUDIT WITH A RETURN CODE OF 12.



                                              Recommendation Summary Report
 Listed below are some solutions to resolving out of synch situations
  that can be flagged within this audit report.

    DUPLIC! (Staging duplicates baseline)
      Delete component from staging or change contents of staging component.
      For package in BAS status, indicates no changes in baseline since pkg installed - no action required.

    SYNCH10! (Version regression problem)
       Copy staging member to development library.
       Checkout member again from the baseline library.
       Resolve version regression.
       Stage member.

CMN7540I - End of job; RC = 12
******************************** BOTTOM OF DATA ********************************************************************
```

ERO Audit for Java

Like package audit, ERO release audit detects source-to-load inconsistencies and out-of-date issues in Java components like it does for non-Java components. There is no source-to-copybook relationship in Java, and the only subroutine-to-composite relationship is between a Java archive (JAR) and its elements. Unlike MVS subroutines in composite loads, JAR elements may be any like-type.

ERO audit analyzes components across release areas and prior releases.

Release Audit Error Numbers for Java

These ERO audit error codes are applicable to JAR components and can be detected by the current ERO release audit.

Error Numbers	Description
100	Identical components
312	Eligible baseline archive element not used by archive
315	Baseline archive element has a more recent date than archive
318	Archive element has a more recent date than archive in baseline
332	Baseline source is later than archive
382	Archive element is not included by baseline archive
401	Archive element has more recent date than archive
404	Source has a more recent date than archive
411	Designated compile procedure not used
413	Archive element deleted
416	Activity file not checked out
417	Version regression
421	Archive element not included in composite archive

For more information, see topic "Release Audit Error Numbers" Chapter 10 "Auditing Release Areas" in the *ChangeMan ZMF ERO Getting Started Guide.*

Known Exception in JAR Relationships

Build procedure CMNJAR includes SYSLIB-type processing to make Java components available for inclusion in a JAR file. However, JAR elements drawn from dependent release areas are not currently being registered in package master subroutine-to-composite relationship records.

Therefore, these audit errors are applicable to JAR components but are not currently detected for and JAR files and JAR elements in dependent release areas.

Error Numbers	Description
310	Eligible dependent area archive element not used by archive
311	Eligible prior release archive element not used by archive
313	Dependent area archive element has a more recent date than archive
314	Prior release archive element has a more recent date than archive
316	Archive element has a more recent date than archive in dependent area
317	Archive element has a more recent date than archive in prior rls
330	Dependent area source is later than archive
331	Prior release source is later than executable
380	Archive element is not included by dependent area archive
381	Archive element is not included by prior release archive

ERO Audit Report for Java Components

This section shows an example of an ERO audit report displaying Java components and relationships.

- Long component names extend across the page, on two lines if necessary.
- Component statistics are shows below the long name line(s).
- JAR files (composite components) are identified by >> in the left margin.
- JAR elements (with no >>) are listed below the JAR name and statistics.

```
      Change Man Release Audit Report       Wednesday May      04,2011   (2011/124) 09:53:49                      PAGE:  13

                ************************(Release Area Processing - zFS Components)*************************
                *Release Identifier     ===> SDZFS001 Created: 20100528   Release Install Date ===> 20100528  *
                *Area Identifier        ===> UNIT                          Area Status          ===> UNBLOCKED *
                *Subcomponent relationships to libtype-(JAR)                                                  *
*****************************************************************************************************************************
*-------------- P r e v i o u s  V e r s i o n --------------*         *---------------- A r e a   L i b r a r y ------------
*********************************************************** *         ***************************************************
>>Component Name -> Subordinate Name
         Timestamp         Size      Area    Release                    Libtype  Appl/Pkg#       Timestamp           Size
          ________________  ________  _______  _______                   _______  __________  ___________________  _________
>>org/jdom/xpath/enh177473jarnew01.jar
                                                                           JAR    STEV000138  2010-05-28 03.22.02       7364
    org/jdom/adapters/package.html
     2010-03-30 07.22.00        193                              ERR0312!    HTH
    org/jdom/adapters/AbstractDOMAdapter.class
     2010-03-26 04.02.22        434                              ERR0312!    JCF
    org/jdom/filter/package.html
     2010-03-30 07.22.13        365                              ERR0312!    HTH
    org/jdom/input/package.html
     2010-03-30 07.22.23        432                              ERR0312!    HTH
    org/jdom/output/package.html
     2010-03-30 07.22.31        610                              ERR0312!    HTH
    org/jdom/package.html
     2010-03-30 07.22.41        524                              ERR0312!    HTH
    org/jdom/transform/package.html
     2010-03-30 07.22.49        282                              ERR0312!    HTH
    org/jdom/xpath/enh177473hthnew01.hth
                                                                  ERR0413!    HTH
    org/jdom/xpath/enh177473hth01.hth
     2010-05-28 02.48.54         90                              ERR0401!    HTH    STEV000138  2010-05-28 05.41.22        137
    org/jdom/xpath/enh177473hth02.hth
     2010-05-28 05.18.33         90                              ERR0315!    HTH
    org/jdom/xpath/enh177473jcfnew01.class
                                                                  ERR0413!    JCF
    org/jdom/xpath/enh177473jcf01.class
     2010-05-26 01.36.50        423                                JCF    STEV000138  2010-05-28 02.44.07        423
    org/jdom/xpath/enh177473jcf02.class
     2010-05-28 05.19.06        422                              ERR0315!    JCF
    org/jdom/xpath/package.html
     2010-03-30 07.22.58        170                              ERR0312!    HTH
    org/jdom/xpath/HelloWorld1.class
     2011-04-15 08.45.00        417                              ERR0315!    JCF
    org/jdom/xpath/HelloWorld3.class
     2010-04-28 07.21.21        417                              ERR0312!    JCF
>>org/jdom/xpath/enh177473jar01.jar
     2010-05-26 03.20.44       6619                                JAR    STEV000138  2010-05-28 03.12.47       6659
    org/jdom/adapters/package.html
     2010-03-30 07.22.00        193                              ERR0312!    HTH
    org/jdom/adapters/AbstractDOMAdapter.class
     2010-03-26 04.02.22        434                              ERR0312!    JCF
    org/jdom/filter/package.html
     2010-03-30 07.22.13        365                              ERR0312!    HTH
    org/jdom/input/package.html
     2010-03-30 07.22.23        432                              ERR0312!    HTH
    org/jdom/output/package.html
     2010-03-30 07.22.31        610                              ERR0312!    HTH
```

Appendix A

Technical Notes

This appendix provides additional technical information about working with Java and zFS in ChangeMan ZMF.

CMNHUTIL - zFS File Utility

CMNHUTIL is a utility program for processing zFS files in batch. It was developed to overcome limitations on path names and file names in z/OS JCL and in dynamic allocation. CMNHUTIL performs these functions:

- Defines a new zFS file
- Copies a single zFS file
- Copies multiple zFS files
- Writes text to a zFS file
- Renames a zFS file
- Deletes a zFS file
- Delete a zFS directory

CMNHUTIL Input

- SYSIN records that specify zFS paths, directories, and files and the functions to be performed
- Input zFS paths, directories, and files specified in SYSIN records

Output

- Output zFS paths, directories, and files specified in SYSIN records
- SYSPRINT report of SYSIN input and processing results

Sample JCL

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

```
//WRJ2TMP EXEC PGM=CMNHUTIL,  copy source files to work files
//             REGION=0M
//SYSPRINT DD  DISP=(,PASS),DSN=&&LIST23,
//             UNIT=SYSDA,SPACE=(133,(1000,1000),RLSE)
//SYSIN    DD  DATA,DLM='++'
CREATEDIRECTORIES=Y
FILEPERMISSIONS=777
FILE=/u/sert/C001/tmp/s6/D63flQV0loG/cmnJVS/jhfjav40.java
define
CREATEDIRECTORIES=Y
FILEPERMISSIONS=777
FILE=/u/sert/C001/tmp/s6/D63flQV0loG/cmngen/jhfjav40.class
DEFINE
CREATEDIRECTORIES=Y
FILEPERMISSIONS=777
FILE=/u/sert/C001/tmp/s6/D63flQV0loG/cmnlst/jhfjav40.list
DEFINE
CREATEDIRECTORIES=Y
FILEPERMISSIONS=777
FILEIN=/cmntp/s6/base/ACTP/stgedev/#000050/JVS/jhfjav40.java
FILEOUT=/u/sert/C001/tmp/s6/D63flQV0loG/cmnJVS/jhfjav40.java
COPY
++
```

DD Statements

This table describes DD statements for CMNHUTIL.

DDNAME	I/O	Purpose

SYSIN	Input	Records that specify zFS paths, directories, and files and the functions to be performed
		IMPORTANT! Code an explicit delimiter ++ for SYSIN in-line data. /* is acceptable in zFS path names, and it will cause truncation if the default in-line data delimiter /* is assumed.
		
		`//SYSIN DD DATA,DLM='++'`
		`. . .`
		`++`

SYSPRINT	Output	Report of SYSIN input and processing results
CMNSDFIL	Output	sidefile records for JAR ILIC use

PARM Options

There are no execution parameters input to utility program CMNHUTIL with a PARM= parameter in the EXEC statement.

SYSIN Parameters

SYSIN records specify zFS paths, directories, and files and the functions to be performed.

There are two types of SYSIN records:

- Keywords - SYSIN keyword statements specify input and output zFS paths, directories, and files. Keyword records also provide parameters and additional information for the function to be performed.
- Verbs - Verb records specify what action is to be taken on the zFS paths, directories, and files specified in SYSIN keyword records.

CMNHUTIL SYSIN Keywords

Keyword statements consist of a keyword and a value. Example:

```
FILEPERMISSIONS=775
```

If a keyword is relevant to an operation and it is omitted, the default value for that keyword is assumed.

This table describes CMNHUTIL SYSIN keywords.

SYSIN Keyword	Description
COMPRESS	Specifies whether output file should be compressed Valid values: Y/N Default: N
CREATEDIRECTORYS	Specifies whether new directory structures should be created Valid values: Y/N Default: N
DIRIN	Specifies input directory Valid values: A valid zFS path name Default: None
DIROUT	Specifies output directory Valid values: A valid zFS path name Default: None
FILE	Specify file for single file operations Valid values: A valid zFS file name Default: None
FILEAPP	Specify single output file for append operations Valid values: A valid zFS file name Default: None
FILEIN	Specify a single input file Valid values: A valid zFS file name Default: None
FILEOUT	Specify a single output file Valid values: A valid zFS file name Default: None
FILEPERMISSIONS	Specifies file permissions for output files Valid values: Valid Unix file permission string Default: 755
INPUTCOMPRESSED	Specifies whether input data file is compressed Valid values: Y/N Default: N

SYSIN Keyword	Description
PRESERVEMODDATE	Specifies whether file modification date should be preserved on copy operations Valid values: Y/N Default: N
REPLACE	Specifies whether files should be replaced in copy operations Valid values: Y/N Default: Y
SIDEFILE	Writes sidefile records to CMNSDFIL Valid values: Y/N Default: N
SUBDIRS	Specifies whether subdirectories should be processed Valid values: Y/N Default: N
TEXTIN	Identifies text for SETTEXT operation Valid values: Any text Default: None

SYSIN Verbs

This table describes CMNHUTIL SYSIN verbs.

SYSIN Verb	Description
COPY	Copy a single file
COPYALL	Copy multiple files
DEFINE	Define a new file
DELETEFILE	Delete a file
RENAMEFILE	Rename a file
RMDIR	Delete a directory
SETTEXT	Write text to file

Valid SYSIN Keyword and Verb Combinations

Not every keyword is relevant to every verb. This table shows valid combinations of keywords and verbs.

	COPY	DEFINE	DELETEFILE	RENAMEFILE	SETTEXT	COPYALL	RMDIR
FILE		X	X				
FILEIN	X			X			
FILEOUT	X			X	X		
FILEAPP	X				X		
DIRIN						X	X
DIROUT						X	
FILEPERMISSIONS	X	X			X	X	

	COPY	DEFINE	DELETEFILE	RENAMEFILE	SETTEXT	COPYALL	RMDIR
CREATEDIRECTORYS	X	X			X	X	
SIDEFILE	X					X	
TEXTIN					X		
SUBDIRS						X	X
COMPRESS	X					X	
INPUTCOMPRESSED	X					X	
PRESERVEMODDATE	X					X	
REPLACE	X					X	

SYSIN Record Syntax Rules

Follow these rules when coding SYSIN keyword statement and verb records.

- A CMNHUTIL operation is defined by one or more keyword statements followed by a verb.
- Each keyword statement or verb is coded on a separate SYSIN record.
- Keyword statements longer than 79 characters are continued by placing a non-blank character in position 80 and continuing the statement in the next SYSIN record, starting in position 1.
- SYSIN may contain multiple keyword statement / verb sets. Operations are performed serially and are initiated as a verb is read.

Return Codes and Error Messages

Utility CMNHUTIL provides two return codes:

- Internal return code displayed in SYSPRINT for each operation (verb), in format

  verb RC:*nnn*

  where nnn is the decimal value a for Unix System Services errno.

  Non-zero internal return codes in SYSPRINT are preceded by an error message that explains the condition.

  For the definition of an errno, see "Return Codes (Errnos) Listed by Value" in the *z/OS UNIX System Services Messages and Codes*.

- External program return code that indicates the overall success or failure of the program. This is the RC and COND CODE listed in the job listing.

  CMNHUTIL considers the internal USS errno in setting the external program return code, but a non-zero USS errno may not be fatal to the execution of CMNHUTIL.

This table describes external return codes for utility CMNHUTIL.

Return Code	Description
00	Successful execution
04	A non-zero internal return code that is not considered fatal to the CMNHUTIL function being executed, or other non-fatal CMNHUTIL conditions.
08	Invalid SYSIN verb or keyword, invalid verb / keyword combination; non-zero internal return code that is considered fatal to the CMNHUTIL function being executed.

Reporting

This is an example of the CMNHUTIL report printed at the SYSPRINT DD statement.

```
*******************************************************************************
* DDNAME: WRJ2TMP.SYSPRINT                                                    *
*******************************************************************************

CMNHUTIL (8.1.0 20141010 14.25) Started...
Keyword processed: CREATEDIRECTORIES                Y
Keyword processed: FILEPERMISSIONS                  777
Keyword processed: FILE                             /u/sert/C001/tmp/s6/D63flQV0loG/cmnJVS/jhfjav40.java
Mkdir:
Mkdir:
define RC: 0000
Keyword processed: CREATEDIRECTORIES                Y
Keyword processed: FILEPERMISSIONS                  777
Keyword processed: FILE                             /u/sert/C001/tmp/s6/D63flQV0loG/cmngen/jhfjav40.class
Mkdir:
DEFINE RC: 0000
Keyword processed: CREATEDIRECTORIES                Y
Keyword processed: FILEPERMISSIONS                  777
Keyword processed: FILE                             /u/sert/C001/tmp/s6/D63flQV0loG/cmnlst/jhfjav40.list
Mkdir:
DEFINE RC: 0000
Keyword processed: CREATEDIRECTORIES                Y
Keyword processed: FILEPERMISSIONS                  777
Keyword processed: FILEIN                           /cmntp/s6/base/ACTP/stgedev/#000050/JVS/jhfjav40.java
Keyword processed: FILEOUT                          /u/sert/C001/tmp/s6/D63flQV0loG/cmnJVS/jhfjav40.java
File: /cmntp/s6/base/ACTP/stgedev/#000050/JVS/jhfjav40.java copied to: /u/sert/C001/tmp/s6/D63flQV0loG/cmnJVS/jhfjav40.ja
va
COPY RC: 0000
```

CMNHUTIL Examples:

- Copy a single file

```
//CMNHUTIL EXEC PGM=CMNHUTIL
//SYSPRINT DD SYSOUT=*
//SYSIN DD DATA,DLM='++'
FILEIN=/u/username/test.txt
FILEOUT=/u/username/backup.txt
COPY
++
```

- Copy an entire directory

```
//CMNHUTIL EXEC PGM=CMNHUTIL
//SYSPRINT DD SYSOUT=*
//SYSIN DD DATA,DLM='++'
DIRIN=/u/username/
DIROUT=/u/usernamebackup/
```

```
FILEPERMISSIONS=777
CREATEDIRECTORIES=Y
SUBDIRS=Y
COPYALL
++
```

Index

Symbols

A

B

C

D

E

F

H

I

J

L

M

O

P

R

S

T

U

W

X

Z