MICRO FOCUS | SERENA

ChangeMan® ZMF

High Level Language Functional Exits Getting Started Guide

Product version: 8.2 Patch 5

Publication date: May 2021

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 provides information that is necessary to create and use exits that you can write in high-level languages such as COBOL, PL/I, and REXX.

This document is intended for ChangeMan ZMF installers, global administrators, and application developers.

Navigating this book

This manual is organized as follows:

- Chapter 1 provides an introduction to functional exits that are written in high-level languages.
- Chapter 2 describes how to set up and implement the high-level language functional exit capability.
- Chapter 3 describes the high-level language interface to ChangeMan ZMF functions such as package create, package update, and build.

Change Bars

Change bars mark the updates to the text in this manual since the last time the manual was updated.

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.
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.

Manual	Description
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 Enterprise Release Option (ERO) of ChangeMan ZMF for managing releases containing change packages.
ERO Getting Started Guide	Explains how to install and use ChangeMan ZMF ERO 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/I, 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 ZMF instance from another CPU or MVS image.
M+R Getting Started Guide	Explains how to install and use the M+R Option of ChangeMan ZMF to consolidate multiple versions of source code and other text components.
M+R Quick Reference	Provides a summary of M+R Option commands in a handy pamphlet format.
Messages	Explains messages issued by ChangeMan ZMF, SERNET, and System Software Manager (SSM) used for the Staging Versions feature of ZMF.
Migration Guide	Gives guidance for upgrading ChangeMan ZMF from Versions 7.1.*x* and 8.1.*x* to Version 8.1.3.
OFM Getting Started Guide	Explains how to install and use the Online Forms Manager (OFM) option of ChangeMan ZMF.

Manual	Description
SER10TY User's Guide	Gives instructions for applying licenses to enable ChangeMan ZMF and its selectable options.
User's Guide	Describes how to use ChangeMan ZMF features and functions to manage changes to application components.
XML Services User's Guide	Documents the most commonly used features of the XML Services application programming interface to ChangeMan ZMF.
ZMF Web Services Getting Started Guide	Documents the Web Services application programming interface to ChangeMan ZMF.

Using the Manuals

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

This section highlights some of the main Reader features. For more detailed information, see the Adobe Reader online help system.

The PDF manuals include the following features:

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

Searching the ChangeMan ZMF Documentation Suite

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

1. Download the ZMF All Documents Bundle ZIP file and the ZMF Readme to your workstation from the My Downloads tab on the Serena Support website.
2. Unzip the PDF files in the ZMF All Documents Bundle into an empty folder. Add the ZMF Readme to the folder.
3. In Adobe Reader, select **Edit** | **Advanced Search** (or press **Shift+Ctrl+F**).

4 Select the **All PDF Documents in** option and use **Browse for Location** in the drop down menu to select the folder containing the ZMF documentation suite.

5 In the text box, enter the word or phrase that you want to find.

6 Optionally, select one or more of the additional search options, such as **Whole words only** and **Case-Sensitive**.

7 Click **Search**.

8 In the **Results**, expand a listed document to see all occurrences of the search argument in that PDF.

9 Click on any listed occurrence to open the PDF document to the found word or phrase.

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

Overview

In order to enforce local business rules, many Serena customers are forced to implement calls to ISPF panel exits from customized versions of the ISPF panel definitions supplied as part of the ZMF product. There are two problems with this:

1. Every time Serena changes a panel definition, or even the underlying code driving the panel display, customers must recustomize and/or retest their local functionality. This consumes a lot of time during a ZMF upgrade.
2. This method of implementing ZMF functionality is only available to the ISPF client. No method is available to ZDD and ZMF for Eclipse clients to achieve the same customization.

ZMF 8.1 and later versions provide central high-level language exit services that can be called by any client that can connect to ZMF. You can code the exits in any Language Environment (LE)-compliant language as well as REXX. The same customer-supplied exit code will be executed regardless of which client is being used.

If an appropriate exit point is in place, there will be no need to code an ISPF panel exit.

NOTE These HLL exit points have no relation to, and do not replace, any existing ZMF assembler exit points (that is, the CMNEX*nnn* exits).

This release addresses requirements in the following ZMF functions:

- Build (including Component Checkin, Stage, Recompile, and Relink)
- Package Create
- Package Update
- File Tailoring
- Component Checkout
- Promote/Demote
- Audit
- Freeze (Unfreeze, Refreeze)
- Approve/Reject
- Revert/Backout
- Package Syslib
- Scratch/Rename
- Issue Reload, Detach, or Attach modify commands

Selected HLL Exits Can Be Coded To Suppress the Display of User Option Panels

The relevant ISPF client programs have been changed to take notice of a proceed=NO setting from the AUDT00UV, BULD00US, BULD00UV, CKOT00UV, FREZ00UV, and PRDM00UV pre-exits. If this setting is found in these exits, the process of displaying the relevant user variable or user options panel or panels is skipped. This change only applies to the only to the ISPF client (and not to ChangeMan ZDD or ChangeMan ZMF for Eclipse client processing).

Getting Started

Detailed descriptions of exit point locations, exit data formats, and task modify commands are provided in the sections which follow. This brief section should be seen as a checklist of how to get started with this feature.

1. HLLX started task procedure - Define the JCL procedure for the HLLX address space in a system procedure library from which it may be started. You should also set up security for this started task to be the same as that for the ZMF/Sernet started task. That is, use the same userid. (See sample JCL member HLLXJCL in the CMNZMF CNTL distribution library. See "The HLL Exit Processing Address Space" on page 18 for an example.)

2. Take a look at the samples provided in the CMNZMF SAMPLES distribution library for the kinds of things that can be done in the HLL exits. The samples should have enough comments for you to be able to understand what they do.

3. Familiarize yourself with the data formats for the different functions. These data formats are described in a later section. Note that there is a standard data structure for each function, regardless of exit point, but that not every exit point has all the information indicated by the full structure.

4. Set up and start your test ZMF 8.1 (or later) subsystem. ZMF will come up without starting the HLLX address space if no active HLL exit points have been defined yet.

5. Connect to the ZMF 8.1 (or later) started task through the ISPF client and use admin option A.G.8 to set your exit values. To begin with you must set the correct name for the HLLX started task procedure. On your first HLLX admin session that's all you need to do. Save this definition away and you will be ready for what comes next.

6. Decide which exit point you are going to investigate first and prepare the exit code. Note that compiled/linked exits are loaded from the HLLX started task STEPLIB. This means you will need access to an APF-authorized library in which to put these exits. This library is concatenated to the standard ZMF/Sernet started task STEPLIB. (This requirement is the same as for the standard ZMF assembler exits.) REXX execs need only be placed in the library referenced by the HLLXREXX ddname in the HLLX started task procedure JCL.

7. Using admin function A.G.8, update the HLLX admin definitions to correctly describe and activate your exit points. When you save (PF3) from the table display the changed contents will be saved to the Package Master.

8. Inform the HLLX address space that the exit definitions have changed. You can use the Administration menu options described in "Modify" on page 128 for all these modify commands. In this case you can issue the following modify command:

   ```
   /F zmfstcname,CMN,ATTACH,HLLX
   ```

   to start the HLLX address space. Note that all modify commands are directed to the ZMF started task, not the HLLX started task.

   The HLLX address space must be notified of any further in-flight changes to exit admin (that is, any changes after the HLLX address space is active) with the following modify command:

   ```
   /F zmfstcname,CMN,RELOAD,HLLX
   ```

Note that all in-flight HLLX conversations are invalidated when you enter this command. All in-flight HLLX conversations are also invalidated if you stop the HLLX address space with the following command:

```
/F zmfstcname,CMN,DETACH,HLLX
```

and restart it again with an ATTACH command (another way of reloading the admin definitions to the HLLX address space). The most intrusive way to reload these definitions is to shut down ZMF (which will automatically shut down HLLX) and restart it.

NOTE The HLLX started task should never be started or stopped directly. It only exists to service requests from ZMF. ZMF starts it up and ZMF shuts it down. It will also respond to an MVS stop command by shutting down, but that should never be necessary in normal operations.

Any displays put in the exits will output in the HLLX address space.

9 Drive the exit(s) by using the relevant ZMF function.

Samples Provided

The CMNZMF SAMPLES distribution library contains a number of members that relate to HLL sample exit code. None of this code is warranted (the exit code belongs to you), but these samples have all been used successfully in testing this feature.

CMNZMF SAMPLES Member Name	Contents
HXC*	Sample HLL exit code written in COBOL.
HXP*	Sample HLL exit code written in PL/I.
HXR*	Sample HLL exit code written in REXX.

Use IEBCOPY or ISPF 3.3 to copy these members to your HLL exit source or REXX exec libraries.

Note that compiled/linked exits *must* be re-entrant.

Copybooks for the exit data areas can be found in the CMNZMF ASMCPY distribution library. They are:

CMNZMF ASMCPY Member Name	Contents
CMNCX*	COBOL copybooks.
CMNPX*	PL/I copybooks.

HLLX Activity Logging

Certain key activities related to HLLX are logged to the standard ZMF log file. There are two activity logging facilities reachable from the ChangeMan ZMF Primary Option Menu (=L).

```
CMNLOGE2                    Select Activity Log Codes       Row 35 to 72 of 81
Command ===> ____________________________________________________ Scroll ===> CSR

    Code  Description
  _ 38    HLLX administration
  _ 39    HLLX commands
```

There are two classes of activity, administrative changes (log indicator 38) and state changes via MVS modify commands (log indicator 39). Any activity which results in an update to the exit definitions in the pmast will result in a log entry which looks like this (when browsing the log file). The modify-from-ISPF facility results in two log entries for each modify command. The ISPF admin function writes a log record as soon as it requests the modify action. Then the modify action itself writes a record. For example, if =A.G.8.Z.1 was used to reload the HLLX exit table it would generate two log records, and then selecting 38 might give a result like this output:

```
CMNLOGDS                      Activity Log Entries             Row 1 to 8 of 8
Command ===> ____________________________________________________ Scroll ===> CSR

Date     Time   User    Package     Description                                 +
20170403 010145 WSER58  HLLX Modfy  Action: RELOAD
20170403 010145 CMNSTAR HLLX cmd:   RELOAD HLLX
****************************** Bottom of data ********************************
```

The general format is the usual log timestamp, followed by the userid making the update, an indication of whether we are updating, modifying or deleting, a procedure or an exit definition. For a procedure definition the only other data in the record is the value of the procedure name after the change has been made. For an exit definition there are a number of values all comma delimited as follows:

Exit: Internal exit name,
Standard exit active Y/N
Standard exit LE or REXX L/R
Debug exit active Y/N
Debug exit LE or REXX L/R
First 8 bytes of standard exit external name
First 8 bytes of debug exit external name
The 10 debug userids

Similarly, selecting 39 might give a result like this output:

```
CMNLOGDS                      Activity Log Entries             Row 1 to 8 of 8
Command ===> ____________________________________________________ Scroll ===> CSR

Date     Time   User    Package     Description                                +
20170322 193834 CMNSTAR HLLX cmd:   RELOAD HLLX
20170322 200835 CMNSTAR HLLX cmd:   RELOAD HLLX
20170322 201120 CMNSTAR HLLX cmd:   RELOAD HLLX
20170322 202529 CMNSTAR HLLX cmd:   RELOAD HLLX
20170322 202843 CMNSTAR HLLX cmd:   RELOAD HLLX
20170323 214011 CMNSTAR HLLX cmd:   RELOAD HLLX
20170323 214851 CMNSTAR HLLX cmd:   DETACH HLLX
20170323 214912 CMNSTAR HLLX cmd:   ATTACH HLLX
****************************** Bottom of data ********************************
```

Note that the userid for all modify commands will be that of the main ZMF started task, in this case CMNSTAR, as the process is actioned by the ZMF main started task.

Chapter 2

High Level Language Exit Details

The HLL Exit Processing Address Space

This address space is started up and shut down by the ZMF/Sernet address space. It can also be restarted, which re-establishes the connection with the ZMF/Sernet address space on an ad-hoc basis.

See "Exit Administration" on page 22 for details on how to specify the name of the HLLX started task.

NOTE The HLLX address space only exists to service requests from a specific instance of the ZMF started task. It must be started by that ZMF started task and never manually. If you start it manually you will see the following messages in the joblog, and the started task will end with RC=20.

```
06.53.45 S0069065 $HASP373 SERDHLL8 STARTED
06.53.46 S0069065 CMNX001E Unable to locate CSA name token.
06.53.46 S0069065 CMNX002E The HLLX a/s must be started only by ZMF.
06.53.46 S0069065 $HASP395 SERDHLL8 ENDED
```

ZMF will only start an HLLX address space if active HLLX exit points have been defined in global administration. (See "Exit Administration" on page 22 for an example of defining HLLX exit points with global administration.)

Member HLLXJCL in the CMNZMF CNTL distribution library provides the JCL to create the HLLX address space. It looks like this:

```
//HLLXPROC PROC PARMS=                          *parms supplied by ZMF
//*
//*  Execute HLL exits in isolation
//*  Started during ZMF initialisation
//*
//*  Note: This STC starts and stops under the control of ZMF *only*
//*        In normal circumstances this stc must not be started or
//*        stopped via MVS operator commands.
//*
//        EXEC PGM=CMNHLLMP,                    *HLL exit monitor pgm
//             PARM='&PARMS',                   *parms
//             REGION=0M,                       *MAXIMUM REGION
//             TIME=NOLIMIT
//*
//STEPLIB   DD DISP=SHR,DSN=somnode.CMNZMF.CUSTOM.LOAD  * Custom Load
//          DD DISP=SHR,DSN=somnode.SERCOMC.CUSTOM.LOAD * Custom Load
//          DD DISP=SHR,DSN=somnode.CMNZMF.LOAD         * Vendor Load
//          DD DISP=SHR,DSN=somnode.SERCOMC.LOAD        * Vendor Load
//*
//HLLXREXX  DD DISP=SHR,DSN=your.rexx.exec.library
//*
//SER#PARM  DD DISP=SHR,DSN=somnode.SERCOMC.TCPIPORT
//SYSTSPRT  DD SYSOUT=X
//SYSPRINT  DD SYSOUT=X
//SYSABEND  DD SYSOUT=X
//SERPRINT  DD SYSOUT=X
//ABNLIGNR  DD DUMMY
//SYSTSIN   DD DUMMY
//SYSIN     DD DUMMY
//*
//* The VPOOL vsam KSDS is optional.
//* See the HLLX Getting Started Guide for further information.
//*
//* Access to the VPOOL vsam dataset may be optimised by making
//* use of a VSAM buffer caching mechanism such as BLSR or SMB.
//* Note that this is unlikely to be critical unless you make
//* heavy use of the VPOOL mechanism.
//*
//* Uncomment the relevant DD statement
//*
//*BLSR:
//*CMNVPALT  DD DISP=SHR,DSN=somnode.CMNZMF.CMNVPOOL
//*CMNVPOOL  DD SUBSYS=(BLSR,'DDNAME=CMNVPALT','STRNO=255')
//*
//*Non-BLSR (to use SMB or no buffering):
//*CMNVPOOL  DD DISP=SHR,DSN=somnode.CMNZMF.CMNVPOOL
```

Notes:

- The TIME=NOLIMIT parameter on the EXEC statement is used to avoid S522 abends due to the fact that the HLL exit processing address space may spend a fair amount of time waiting for work.
- The HLLXREXX ddname identifies the exec library for REXX execs.
- The CMNVPOOL ddname is optional. It refers to the VSAM data set that stores the variables that the exits use. See "Variable Pool Function - CMNVPOOL" on page 37 for more information about CMNVPOOL.
- The SER#PARM ddname is only needed if the exits will be invoking ZMF XML services (and then it is only really needed to avoid the ddname missing warning message).
- You should also add any libraries/files that your exit code requires.

Here are some typical messages seen in ZMF SERPRINT when active HLL exits exist at the time ZMF starts up:

```
2014/03/26 06:51:11.98 CMN8401I CMNSTART Waiting for HLL exit address space to initialize.
2014/03/26 06:51:12.55 CMN8460I CMNDELAY - START of processing
2014/03/26 06:51:12.55 CMN8468I CMNDELAY - Waiting for resource => qname:rname
2014/03/26 06:51:12.55 CMN8468I CMNDELAY - CHGMAN:CMNDEV.CMNSYS.U810STEV.CMNDELAY
2014/03/26 06:51:12.55 CMN8469I CMNDELAY - Resource obtained successfully.
2014/03/26 06:51:12.68 CMN8800I SERD     Opened VSAM file CMNDEV.CMNSYS.U810STEV.CMNDELAY     Exclusiv
2014/03/26 06:51:12.80 CMN8800I SERD     Closed VSAM file CMNDEV.CMNSYS.U810STEV.CMNDELAY
2014/03/26 06:51:12.80 CMN8461I CMNDELAY - END of processing                    00000
2014/03/26 06:51:20.99 CMN8402I CMNSTART Successfully connected to HLL exit address space.
2014/03/26 06:51:20.99 CMN8413I Start of HLLX active exits list:
2014/03/26 06:51:20.99 CMN8415I IntName  Typ Env  External Name        Debug Ids
2014/03/26 06:51:20.99 CMN8416I PCRE0007 STD REXX GENPCRE
2014/03/26 06:51:20.99 CMN8416I PCRE0107 STD REXX DEFLTSIT
2014/03/26 06:51:21.00 CMN8414I End of HLLX active exits list.
2014/03/26 06:52:00.10 SER0868I EPvt used=19,964K avail=1,714,692K  Pvt used= 220K avail=8,972K
2014/03/26 06:52:00.10 CMN8305I CMNSTART Detach: U=CMNSTART,F1=DETACH,F2=SERVDLAY,@TCA=17574000,@TCB=8B2298
2014/03/26 06:52:00.11 CMN8303I CMNSTART Initialization Complete
```

Initialization, Termination, Modify Commands

During startup ZMF needs to decide whether the HLLX address space is needed at all. CMNSTART will read the Package Master for the HLL exit admin records. If no HLL exit points are active, you need to do nothing to activate the HLLX support. The following message is issued:

```
CMN_418I CMNSTART No active HLL exits are defined, the HLLX address
    space will not be started.
```

Other messages you may see:

```
CMN_401I CMNSTART Waiting for HLL exit address space to initialize.
CMN_402I CMNSTART Successfully connected to HLL exit address space.
CMN_409I CMNSTART Prior instance of HLLX still active, please try later.
CMN_410I CMNSTART HLL exit address space creation failed.
CMN_411I CMNSTART HLL exit address space failed to initialize, re-
    attempt via ATTACH command.
```

You will also see the response to an internally issued DISPLAY HLLX command (see below) which shows all active HLL exit points.

During ZMF shutdown you may see:

```
CMN_403I CMNSTART Termination of HLL exit address space requested.
CMN_404I CMNSTART Termination of HLL exit address space complete.
CMN_405I CMNSTART Nothing heard from HLLX, unilateral termination.
```

The following ZMF/Sernet modify commands are available:

```
F zmfstcname,CMN,DETACH,HLLX

F zmfstcname,CMN,ATTACH,HLLX
```

Detaching HLLX will ensure that the existing (known) HLLX started task is either gone or request it to go away. We will then follow what is left of the ZMF termination process for

HLLX listed above so that we are in a state as if we were initializing the HLLX address space during ZMF startup. You may need to do this if something has gone awry with the HLLX process and you are locked out of ZMF.

In response to the attach request we will go through the startup process for HLLX. If HLLX is found to be already active, we will issue the general message 315 (this message is already coded in CMNSTART and takes different text depending on which subtask is being attached):

```
CMN_315I CMNSTART HLLX address space is already active, no action taken.
```

If no active HLL exits are defined, the HLLX started task is not started and the following message is issued (similar to startup):

```
CMN_418I CMNSTART No active HLL exits are defined, the HLLX address
    space will not be started.
```

A further operator command:

```
F zmfstcname,CMN,RELOAD,HLLX
```

will result in the active HLLX exits table being refreshed from the definitions saved in the Package Master:

```
CMN_406I CMNSTART HLLX active exit table has been reloaded.
```

This is followed by a list of all active HLL exits, as per the DISPLAY command documented below.

If the reloaded table contains no active exits, there is no need for further HLLX support and the process will request shutdown of HLLX support. This can be reactivated by updating the exits admin and then issuing an ATTACH command or restarting ZMF.

```
CMN_407I CMNSTART No active HLLX exits are defined, HLLX will shut down.
CMN_408I CMNSTART Please issue ATTACH HLLX command or restart ZMF if you
    wish to re-activate one or more exits.
```

There is also a command to display which exits are currently active in the ZMF/HLLX set up:

```
F zmfstcname,CMN,DISPLAY,HLLX
```

If no exits are active you will see this message:

```
CMN_412I There are currently no active HLL exits.
```

Otherwise, this set of messages will be produced, a line for each active exit:

```
CMN_413I Start of HLLX active exits list:
CMN_415I IntName  Typ Env  External Name        Debug Ids
CMN_416I PCRE0001 STD REXX PNL01PRE
CMN_417I PCRE0001 DBG REXX TST01PRE             WSER58  ,SDOWNES
CMN_416I PCRE0101 STD REXX PNL01PST
CMN_417I PCRE0101 DBG REXX TST01PST             WSER58  ,SDOWNES
CMN_416I PCRE0007 STD  LE  PLI7PRE
CMN_416I PCRE0107 STD REXX PNL07PST
CMN_416I PUPD0002 STD REXX GENPUPD
CMN_416I PUPD0102 STD REXX GENPUPD
CMN_416I PUPD0003 STD  LE  PUPDGNL
CMN_416I PUPD0103 STD  LE  PUPDGNL
CMN_416I PUPD0004 STD REXX GENPUPD
CMN_416I PUPD0104 STD REXX GENPUPD
CMN_416I PUPD0005 STD  LE  PUPDGNL
CMN_416I PUPD0105 STD  LE  PUPDGNL
CMN_416I PUPD0006 STD REXX GENPUPD
CMN_416I PUPD0106 STD REXX GENPUPD
CMN_414I End of HLLX active exits list.
```

The 416 message is issued for each active standard exit definition and shows:

- The internal exit name.
- The fact that it is a standard exit definition.
- Whether it is REXX or Language Environment (LE).
- The external module name that implements the functionality.

The 417 message is issued for each active debug exit definition and shows, in addition to the 416 information, the list of up to 10 userids that will take the debug exit ahead of the standard exit.

Exit Administration

The HLLX process needs to know certain information about the various HLL exit points. This information includes:

- The program/exec name that you wish to use for each exit point.
- Whether the defined exit point is active or not.
- Whether the exit is an LE-language program or a REXX exec.

In addition, a debug mechanism has been implemented whereby certain users will be able to take an alternatively named exit module to allow isolated testing of exit changes. Thus, we also have an alternative program/exec name which will be taken by a list of userids that will take this debug version of the exit.

Administration of the exit points is also where you let ZMF know the name of the procedure you wish to be started to service the HLLX requests. Here's a sample Update Global Administration Options (CMNGAMN1) panel:

```
CMNGAMN1             Update Global Administration Options
Option ===>

1  Parms          Global parameters
2  Library        Library types
3  Language       Language names
4  Procedures     Compiling procedures
5  Reason Codes   Reason codes for unplanned packages
6  Sites          Site information
7  Lock           Application parameter locks
8  HLL Exits      High level language exits
9  Field Names    User field name substitution
C  Component      Component information
D  Dates          Installation calendar
E  REST           REST api server
H  Housekeeping   Housekeeping tasks
I  Impact         Impact Analysis
N  Notify         Global notification file
O  Options        Selectable options
R  Reports        ChangeMan ZMF batch reports
S  Skeletons      Skeleton procedures
```

If you are on a P-Site, the CMNGAMN2 panel appears instead:

```
CMNGAMN2             Update Global Administration Options
Option ===> ____________________________________________

1  Parms          Global parameters
7  Lock           Application parameter locks
8  HLL Exits      High level language exits
9  Field Names    User field name substitution
D  Dates          Installation calendar
H  Housekeeping   Housekeeping tasks
N  Notify         Global notification file
O  Options        Selectable options
R  Reports        ChangeMan ZMF batch reports
```

The HLL Exit Definition - Function Selection (CMNHLLMM) panel allows administrators to choose to work with all the definitions or just those for a specific function. (More functions will be added as Serena expands this facility.):

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

On a P-Site, the panel CMNHLLPS appears:

```
CMNHLLPS           HLL Exit Definition (P-Site)- Function Selection
Option ===> ____________________________________________________

R  Revert/Backout  Package revert and backout
M  Miscellaneous   HLLX procedure name

Z  Modify          Issue Reload, Detach, or Attach modify commands
```

Option M allows you to define the HLLX procedure name:

```
CMNHLLMP              HLL Exit Miscellaneous Parameters
Command ===> ____________________________________________ Scroll ===> CSR

 HLLX procedure name . . SERDHLLR
```

Note that the variable on this panel is the name of the HLLX started task procedure. This name must be specified correctly before you attempt to activate any exits. If you run into problems with this (for example, the procedure name is wrong, or the procedure fails) with exits being active, you may find you are locked out of ZMF. If this happens, you will need to detach the HLLX subtask with the following command (or use the Modify menu option):

```
/F zmfstcname,CMN,DETACH,HLLX
```

The following example shows the selection of option 3 to display the package-create exit points:

```
CMNHLLMN                      HLL Exit Definition           Row 5 to 10 of 10
Command ===> ____________________________________________________ Scroll ===> CSR

Internal External  + Active  1=LE    Description  +
  Name     Name              2=REXX  Debug Userids  +
-------------------------------------------------------------------------------
PCRE0001 PNL01PRE     YES    2       Pre 1st panel for pkg create
 Debug:  TST01PRE     NO     2       WSER58
-------------------------------------------------------------------------------
PCRE0101 PNL01PST     YES    2       Post 1st panel for pkg create
 Debug:  PNL01PST     NO     2       WSER58
-------------------------------------------------------------------------------
PCRE0006 PNL07PRE     YES    1       Pre final panel for ALL site PCRT
 Debug:  ________      NO     2
-------------------------------------------------------------------------------
PCRE0106 PNL07PST     YES    2       Post final panel for ALL site PCRT
 Debug:  ________      NO     2
-------------------------------------------------------------------------------
PCRE0007 PNL07PRE     YES    1       Pre final panel for DP site PCRT
 Debug:  ________      NO     2
-------------------------------------------------------------------------------
PCRE0107 PNL07PST     YES    2       Post final panel for DP site PCRT
 Debug:  ________      NO     2
```

You can use a Locate command with the full internal exit name on the table displays.

Each defined exit point has a fixed internal name (for example, PCRE0001 for package create *pre*-panel 01, PCRE0101 for package create *post*-panel 01). We relate these fixed names to the customer-preferred program/exec names.

There are two lines per exit, one for the standard definition and one for the debug definition. When the HLLX subtask decides which to load it checks the incoming userid against the list of debug userids from these records as stored in the active exits table.

The defaults for exits not defined in the Package Master will be to make them inactive with no debug module or userid list.

Caller to Exit Cross Reference

HLL exits are called either by client code or by service code.

The service exits all have X in the seventh position of the exit point name, for example, BULD00XC, and are called regardless of where the original request for the service came from.

There are four categories of "caller" which are identified to each exit in the callOrigin field:

- SPF - the exit has been called by an ISPF client function. That function can take all client exits defined for its functional path and the service exits for whichever service the function eventually drives. The list of exits driven will differ from one function to the next.
- ZDD - the exit has been called by a ZDD client function. Note that there is no one-to-one correlation between ISPF client processing and any other client processing. There

could be (and usually is) a completely different way of presenting the function to the user in each client. In some areas the correlation is close, in others it is not. Please refer to the table that follows for a list of which client exits are called by ZDD. Note that any services driven by the ZDD client will also take the relevant service defined HLL exit points.

- ECL - the exit has been called by a ZMF4ECL (ZMF for Eclipse) client function. The same comments as made for ZDD apply to ZMF4ECL.
- XML - anything else that drives a ZMF XML service (so, for example, direct service invocation from custom REXX execs or REST API calls). ChangeMan ZMF has no control over client code in these cases and so only the service-oriented exits will be taken. You may implement whatever business logic you want in the service exits and know that it will be driven regardless of which client invoked the service.

The client exits allow you to modify the information gathered from, and actions precipitated by, the normal flow of the three clients. To re-emphasize, there is no one-to-one correlation between the three clients function flows.

To apply business logic across the board (regardless of which client is being used) then the relevant service exit can be used.

However, the service is usually only invoked at the end of a functional process and if custom validation throws out the service request it may be frustrating for a client user to be told this only after progressing through many panels/windows of data gathering.

Thus, the client exits are there to allow "real time" validation (etc.) to be performed.

The service exits are informed which of the recognized categories have invoked this service (i.e. callOrigin will be one of "SPF", "ZDD", "ECL", "XML") so that, if you have implemented client exits to do certain validation (etc.) then you needn't repeat this processing again in the service exit. You can implement logic so that the same validation is only performed if the origin of the service request is identified as "XML" (for example).

The following table lists all the currently available HLL exits along with check columns to show who takes them.

Internal exit name	Short description	Service exit	Driven by		
			ISPF	ZDD	ZMF4ECL
APRV00XM	pre service call for approve	Y	n/a	n/a	n/a
APRV01XM	post service call for approve	Y	n/a	n/a	n/a
APRV0101	list of packages to be approved		Y		
APRV0102	approve/reject option menu		Y		
APRV0103	approver entity list		Y	Y	Y
APRV0004	pre checkoff comments		Y	Y	Y
APRV0104	post checkoff comments		Y	Y	Y
APRV0105	reject reasons entity selection		Y		
APRV0006	pre reject reasons text		Y	Y	Y
APRV0106	post reject reasons text		Y	Y	Y
AUDT00AR	pre autoresolve job submission		Y		

Internal exit name	Short description	Service exit	Driven by		
			ISPF	ZDD	ZMF4ECL
AUDT01JB	post all audit job processing		Y		
AUDT00RC	pre audit job setting of package RC		Y		
AUDT00UV	pre user variables		Y	Y	Y
AUDT01UV	post user variables		Y	Y	Y
AUDT00XM	pre audit job submission service	Y	n/a	n/a	n/a
AUDT01XM	post audit job submission service	Y	n/a	n/a	n/a
AUDT0001	pre audit submission panel		Y	Y	Y
AUDT0101	post audit submission panel		Y	Y	Y
AUDT0002	pre application in scope panel		Y	Y	Y
AUDT0102	post application in scope panel		Y	Y	Y
BULD01DL	post cmpnt delete selection		Y	Y	
BULD01LO	post relink output library selection		Y		Y
BULD01LR	post relink library selection		Y		Y
BULD01LT	post relink member list		Y		
BULD00L0	pre initial relink (rebind) panel		Y		Y
BULD01L0	post initial relink (rebind) panel		Y		Y
BULD00L1	pre relink job submission		Y	Y	Y
BULD01L1	post relink job submission		Y	Y	Y
BULD01RC	post recompile confirmation		Y		
BULD01RL	post recompile library selection		Y		
BULD01RP	post promotion library selection		Y		
BULD01RR	post release library selection		Y		
BULD00R0	pre initial recompile panel		Y		
BULD01R0	post initial recompile panel		Y		
BULD00R1	pre standard recompile submission		Y	Y	Y
BULD01R1	post standard recompile submission		Y	Y	Y
BULD01R2	post recompile component list		Y		
BULD00R3	pre mass/batch recompile submission		Y	Y	Y
BULD01R3	post mass/batch recompile submission		Y	Y	Y
BULD00US	pre component user variables		Y	Y	Y
BULD01US	post component user variables		Y	Y	Y
BULD00UV	pre user variables		Y	Y	Y
BULD01UV	post user variables		Y	Y	Y
BULD00XB	pre BUILD xml service	Y	n/a	n/a	n/a

Internal exit name	Short description	Service exit	Driven by		
			ISPF	ZDD	ZMF4ECL
BULD01XB	post BUILD xml service	Y	n/a	n/a	n/a
BULD00XC	pre CHECKIN xml service	Y	n/a	n/a	n/a
BULD01XC	post CHECKIN xml service	Y	n/a	n/a	n/a
BULD00XD	pre DELETE xml service	Y	n/a	n/a	n/a
BULD01XD	post DELETE xml service	Y	n/a	n/a	n/a
BULD00XL	pre RELINK xml service	Y	n/a	n/a	n/a
BULD01XL	post RELINK xml service	Y	n/a	n/a	n/a
BULD00XR	pre RECOMPILE xml service	Y	n/a	n/a	n/a
BULD01XR	post RECOMPILE xml service	Y	n/a	n/a	n/a
BULD0101	post package component list		Y	Y	
BULD0002	pre stage from dev initial panel		Y	Y	Y
BULD0102	post stage from dev initial panel		Y	Y	Y
BULD0103	post Stage from dev member selection		Y	Y	Y
BULD0004	pre standard stage job submission		Y	Y	Y
BULD0104	post standard stage job submission		Y	Y	Y
BULD0005	pre mass stage job submission		Y	Y	Y
BULD0105	post mass stage job submission		Y	Y	Y
BULD0106	post procedure selection		Y		
BULD0107	post language selection		Y		
BULD0108	post library type selection		Y		
BULD0009	pre Like-O std stage job submission		Y	Y	Y
BULD0109	post Like-O std stage job submission		Y	Y	Y
BULD0010	pre Like-O mass stg job submission		Y		Y
BULD0110	post Like-O mass stg job submission		Y		Y
BULD0111	post recfm=U dev stage mbr selection		Y		
BULD0012	pre component selection parameters		Y		
BULD0112	post component selection parameters		Y		
BULD0015	pre component general description		Y		
BULD0115	post component general description		Y		
BULD0117	post valid staging line commands		Y		
BULD0118	post DB2 subsystem selection		Y		
BULD0019	pre batch stage jobcard definition		Y		
BULD0119	post batch stage jobcard definition		Y		
BULD0123	post dev stg hfs file selection		Y		Y

Internal exit name	Short description	Service exit	Driven by		
			ISPF	ZDD	ZMF4ECL
BULD0025	pre extract SP from DB2 panel		Y	Y	
BULD0125	post extract SP from DB2 panel		Y	Y	
BULD0026	pre mass rebuild job submission		Y		
BULD0126	post mass rebuild job submission		Y		
CKOT01CK	post checkout entry panel		Y		
CKOT01CL	post component list for pkg ckot		Y		
CKOT01DL	post component delete		Y		
CKOT01LB	post library list		Y		
CKOT01LT	post libtype display		Y		
CKOT01MS	post member selection list		Y	Y	Y
CKOT01PL	post promotion library list		Y		
CKOT00UV	pre user variables		Y	Y	Y
CKOT01UV	post user variables		Y	Y	Y
CKOT00XM	pre service call	Y	n/a	n/a	n/a
CKOT01XM	post service call	Y	n/a	n/a	n/a
CKOT0001	pre checkout selection criteria		Y	Y	Y
CKOT0101	post checkout selection criteria		Y	Y	Y
CKOT0002	pre batch checkout panel		Y	Y	
CKOT0102	post batch checkout panel		Y	Y	
CKOT0003	pre checkout from package entry		Y		
CKOT0103	post checkout from package entry		Y		
FREZ00UF	pre package freeze/unfreeze panel		Y		
FREZ01UF	post package freeze/unfreeze panel		Y	Y	
FREZ00UV	pre user variables		Y	Y	
FREZ01UV	post user variables		Y	Y	Y
FREZ01U1	selective component freeze/unfreeze		Y	Y	Y
FREZ01U2	selective utility freeze/unfreeze		Y		
FREZ00XM	pre package freeze service	Y	n/a	n/a	n/a
FREZ01XM	post package freeze service	Y	n/a	n/a	n/a
FREZ00XR	pre selective refreeze service	Y	n/a	n/a	n/a
FREZ01XR	post selective refreeze service	Y	n/a	n/a	n/a
FREZ00XU	pre selective unfreeze service	Y	n/a	n/a	n/a
FREZ01XU	post selective unfreeze service	Y	n/a	n/a	n/a
FREZ0101	package freeze submenu		Y	Y	Y

Internal exit name	Short description	Service exit	Driven by		
			ISPF	ZDD	ZMF4ECL
FREZ0002	pre batch freeze submit panel		Y	Y	Y
FREZ0102	post batch freeze submit panel		Y	Y	Y
FTLR00BA	install/baseline file tailoring		Y		
FTLR00BL	build job file tailoring		Y		
FTLR00BP	base ZMF promotion file tailoring		Y		
FTLR00EB	ERO autoresolve file tailoring		Y		
FTLR00EP	ERO promotion file tailoring		Y		
PCRE00PU	pre package user options		Y	Y	Y
PCRE01PU	post package user options		Y	Y	Y
PCRE00XM	pre package create xml service	Y	n/a	n/a	n/a
PCRE01XM	post package create xml service	Y	n/a	n/a	n/a
PCRE0001	pre initial pkg create panel		Y	Y	Y
PCRE0101	post initial pkg create panel		Y	Y	Y
PCRE0002	pre package description		Y	Y	Y
PCRE0102	post package description		Y	Y	Y
PCRE0003	pre installation instructions		Y	Y	Y
PCRE0103	post installation instructions		Y	Y	Y
PCRE0004	pre scheduling dependencies		Y	Y	Y
PCRE0104	post scheduling dependencies		Y	Y	Y
PCRE0005	pre affected applications		Y	Y	Y
PCRE0105	post affected applications		Y	Y	Y
PCRE0006	pre install information (ALL site)		Y	Y	Y
PCRE0106	post install information (ALL site)		Y	Y	Y
PCRE0007	pre install site list (DP site)		Y	Y	Y
PCRE0107	post install site list (DP site)		Y	Y	Y
PCRE0008	pre complex/super information		Y	Y	Y
PCRE0108	post complex/super information		Y	Y	Y
PRDM00UV	pre user variables		Y	Y	Y
PRDM01UV	post user variables		Y	Y	Y
PRDM00XD	pre demotion service	Y	n/a	n/a	n/a
PRDM01XD	post demotion service	Y	n/a	n/a	n/a
PRDM00XP	pre promotion service	Y	n/a	n/a	n/a
PRDM01XP	post promotion service	Y	n/a	n/a	n/a
PRDM0100	post promote/demote main menu		Y	Y	

Internal exit name	Short description	Service exit	Driven by		
			ISPF	ZDD	ZMF4ECL
PRDM0101	post site selection		Y		
PRDM0003	pre promote options		Y	Y	Y
PRDM0103	post promote options		Y	Y	Y
PRDM0004	pre demote options		Y	Y	Y
PRDM0104	post demote options		Y	Y	Y
PRDM0105	post selective promote/demote		Y	Y	Y
PRDM0107	post promotion level selection		Y		
PUPD00D2	pre DB2 option special libtypes		Y		
PUPD01D2	post DB2 option special libtypes		Y		
PUPD00ER	pre ERO information		Y		
PUPD01ER	post ERO information		Y		
PUPD00IA	pre IMS option ACB definitions		Y		
PUPD01IA	post IMS option ACB definitions		Y		
PUPD00ID	pre IMS option DBD definitions		Y		
PUPD01ID	post IMS option DBD definitions		Y		
PUPD00IP	pre IMS option PSB definitions		Y		
PUPD01IP	post IMS option PSB definitions		Y		
PUPD00IS	pre IMS option physical systems		Y		
PUPD01IS	post IMS option physical systems		Y		
PUPD01M6	post monitor pkg change insdate(ALL)		Y		
PUPD01M7	post monitor pkg change insdate (DP)		Y		
PUPD00PU	pre package user options		Y	Y	Y
PUPD01PU	post package user options		Y	Y	Y
PUPD0001	pre control information		Y	Y	Y
PUPD0101	post control information		Y	Y	Y
PUPD0002	pre package description		Y	Y	Y
PUPD0102	post package description		Y	Y	Y
PUPD0003	pre installation instructions		Y	Y	Y
PUPD0103	post installation instructions		Y	Y	Y
PUPD0004	pre scheduling dependencies		Y	Y	Y
PUPD0104	post scheduling dependencies		Y	Y	Y
PUPD0005	pre affected applications		Y	Y	Y
PUPD0105	post affected applications		Y	Y	Y
PUPD0006	pre install information (ALL site)		Y	Y	Y

Internal exit name	Short description	Service exit	Driven by		
			ISPF	ZDD	ZMF4ECL
PUPD0106	post install information (ALL site)		Y	Y	Y
PUPD0007	pre install site list (DP site)		Y	Y	Y
PUPD0107	post install site list (DP site)		Y	Y	Y
PUPD0008	pre complex/super information		Y	Y	Y
PUPD0108	post complex/super information		Y	Y	Y
RCKI01AR	post-CI line cmd for area checkin		Y		
RCKI01CI	post-action call for ERO checkin		Y		
RCKI01PK	post-CI line cmd for pkg checkin		Y		
RCKI00XM	pre-service call for ERO checkin	Y	n/a	n/a	n/a
RCKI01XM	post-service call for ERO checkin	Y	n/a	n/a	n/a
RCKI0000	pre-pkg checkin options		Y	Y	
RCKI0100	post-pkg checkin options		Y	Y	
RCKI0001	pre-pkg checkin selection parms		Y		
RCKI0101	post-pkg checkin selection parms		Y		
RCKI0102	post-pkg eligible components list		Y	Y	
RCKI0103	post-pkg duplicate components list		Y	Y	
RCKI0107	post-summary of pkg checkin		Y	Y	
RCKI0020	pre-component description checkin		Y		
RCKI0120	post-component description checkin		Y		
RCKI0050	pre-area checkin options		Y	Y	
RCKI0150	post-area checkin options		Y	Y	
RCKI0051	pre-area checkin selection parms		Y		
RCKI0151	post-area checkin selection parms		Y		
RCKI0152	post-area eligible components list		Y	Y	
RCKI0153	post-area duplicate components list		Y	Y	
RCKI0157	post-summary of area checkin		Y	Y	
RCKI0070	pre-batch checkin job submission		Y		
RCKI0170	post-batch checkin job submission		Y		
RVRT01B1	package backout selection panel		Y		Y
RVRT00B2	pre backout reason entry panel		Y		Y
RVRT01B2	post backout reason entry panel		Y		Y
RVRT01B3	backout site selection		Y		Y
RVRT00B4	pre backout remote submission		Y		Y
RVRT01B4	post backout remote submission		Y		Y

Internal exit name	Short description	Service exit	Driven by		
			ISPF	ZDD	ZMF4ECL
RVRT00XB	pre service call for backout	Y	n/a	n/a	n/a
RVRT01XB	post service call for backout	Y	n/a	n/a	n/a
RVRT00XM	pre service call for revert	Y	n/a	n/a	n/a
RVRT01XM	post service call for revert	Y	n/a	n/a	n/a
RVRT0101	package revert selection panel		Y	Y	Y
RVRT0002	pre revert reason entry panel		Y	Y	Y
RVRT0102	post revert reason entry panel		Y	Y	Y
RVRT0103	revert site selection		Y	Y	Y
RVRT0004	pre revert remote submission		Y	Y	Y
RVRT0104	post revert remote submission		Y	Y	Y
SCRN00XM	pre service call for pkg_util	Y	n/a	n/a	n/a
SCRN01XM	post service call for pkg_util	Y	n/a	n/a	n/a
SCRN0101	post package selection		Y		Y
SCRN0002	pre baseline selection		Y	Y	
SCRN0102	post baseline selection		Y	Y	Y
SCRN0103	post baseline member list		Y		
SCRN0104	post package member list		Y	Y	
SCRN0105	post libtype selection list		Y		Y
SYSL00XL	pre srvc cmponent.listlang.history	Y	n/a	n/a	n/a
SYSL01XL	post srvc cmponent.listlang.history	Y	n/a	n/a	n/a
SYSL00XM	service call for package.list.syslib	Y	n/a	n/a	n/a
SYSL00XR	service call->package.refresh.syslib	Y	n/a	n/a	n/a

ISPF or TSO in HLL Exits is unsupported

The starting of an ISPF environment or the use of ISPF services from within an HLL exit is not supported. If you try to do this you will run into unpredictable results. Similarly a TSO environment is not supported. An example of when you might want to use TSO is to submit a batch job (one way is to use BPXWDYN for dynamic allocations, and use the internal reader - you could use REXX to build up a stem variable, and EXECIO).

Sample LE-Language Exit Modules

A limited number of sample exit programs are provided based on those used for testing. These samples show basic operations such as field preparation and validation along with

more involved processes such as executing ZMF XML requests, making Db2 requests, file allocation and reading, and use of the CMNVPOOL facility.

The samples are available in the CMNZMF SAMPLES distribution library with names HXC* (COBOL) and HXP* (PL/I).

Note that all LE exits and all called subroutines *must* be reentrant. It is also required that you use ALL31(ON) to ensure that LE HLL exit processing is able to support a high level of concurrency. Error messages are sent to the HLLXMSG ddname. Trace output is sent to TRCnnnnn ddnames.

Once you've seen how one exit point works, that knowledge can be applied to any of them.

In general, these samples are neither guaranteed nor supported. The specific items that Serena supports are:

- The format of the data passed to the exit.
- If any of the data is changed, the function in progress will pick up those changes and act on them.
- Use of LE GETMAIN services to extend the length of passed variable lists.
- Use of ZMF XML services from within the exits.
- Use of the CMNVPOOL facility.

NOTE Compiled REXX routines are supported in exactly the same manner as standard REXX routines. However, REXX routines built as standard load modules using MVS stub functionality are neither REXX nor LE-compliant. Hence they are not supported in HLLX.

The following code snippets explain how LE languages process the variable blocks in the package-create function. The data interface for the high-level language exits is given for each functional area in Chapter 3, "ZMF/HLL Exit Interface" on page 45. For COBOL:

```
       S500-VARIABLE-BLOCK.
      *********************
      *             * NOW PROCESS EACH VARIABLE  *
      *             * BLOCK THAT MAY HAVE BEEN   *
      *             * PASSED TO US               *
      *             ******************************
      *             * CALLED FROM:               *
      *             * S500-VARIABLE-BLOCK        *
      *             ******************************
           PERFORM S510-PROCESSVB1 UNTIL VB1DONE.
.
.
.
        S510-PROCESSVB1.
       ******************
       *             * DISPLAY VARIABLES FROM THE *
       *             * FIRST PARM BLOCK           *
       *             * THEN USE THE NEXT POINTER  *
       *             * TO GET ADDRESSABILITY TILL *
       *             * THAT POINTER IS NULL TO    *
       *             * DENOTE END OF LINKED LIST  *
       *             ******************************
       *             * CALLED FROM:               *
       *             * S500-VARIABLE-BLOCK        *
       *             ******************************
            DISPLAY 'PACKAGE DESCRIPTION : ' PCRTPDSC.
            IF PTR-NEXT-PCRTVB1 NOT = NULLS
                 SET ADDRESS OF PCRTVB1 TO PTR-NEXT-PCRTVB1
            ELSE
               MOVE "Y" TO WS-VB1DONE
            END-IF.
```

For PL/I, using the same data structure:

```
*PROCESS NAME('PLI7PRE') INCLUDE MARGINS(2,72,1) OPTIMIZE(TIME);
1PLI7PRE:PROC(PCRT) OPTIONS (ASM);

 DCL  SYSNULL           BUILTIN;
 DCL  DESC_DONE         CHAR(1);

 /* API data fields layout */
    %INCLUDE CMNPXPCR;
.
.
.
 IF PCRTVB1P ¬= SYSNULL() THEN DO;
    DESC_DONE = 'N';
    WORKVB1P  =  PCRTVB1P;

    DO WHILE (DESC_DONE = 'N');
       PUT SKIP LIST('DESC LINE: '||PCRTPDSC);
       IF PTR_NEXT_PCRTVB1 = SYSNULL() THEN
          DESC_DONE = 'Y';
       ELSE
          WORKVB1P  =  PTR_NEXT_PCRTVB1;
    END;
 END;
```

Sample REXX Execs

Similar comments apply to sample REXX execs. Sample REXX execs are in the CMNZMF SAMPLES distribution library with names HXR*.

Note that to avoid clashes in variable names when you call SERXMLRC for ZMF services, ensure that your stem variable name prepends something prior to the tagname. For example:

```
stem = 'HLL7.ZMF_'
drop HLL7.
HLL7.                              = ""
HLL7.ZMF_Subsys                    = zmfSubs
HLL7.ZMF_Userid                    = userid
HLL7.ZMF_Test                      = " "
HLL7.ZMF_Product                   = "CMN"
HLL7.ZMF_lproduct                  = "ZMF"
HLL7.ZMF_Service                   = "PARMS"
HLL7.ZMF_Message                   = "LIST"
HLL7.ZMF_Scope                     = "APL"
HLL7.ZMF_applName                  = applName

HLL7.ZMF_includeInResult.1         = "applDesc"

address LINKMVS "SERXMLRC stem"
```

Variable Pool Function – CMNVPOOL

The purpose of this (optional) function is to provide a simple method of saving and accessing variable data across HLL exit execution. The function can be called from either LE programs (using the CMNLPOOL front end) or from REXX execs (using CMNRPOOL). Examples are given below.

Sample JCL to allocate the CMNVPOOL data set is provided in the HLLXVPL member of the CMNZMF.V8R1M4.CNTL distribution library.

The repository for customer-defined variable data is a VSAM KSDS, which is defined in a similar manner to the following:

```
//*
//IDCAMS  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//DUMMY     DD *
DUMMY RECORD
//SYSIN     DD *
 DELETE CMNDEV.CMNSYS.U810ALL.CMNVPOOL
 SET MAXCC = 0
 /*
 /* Delete, Define, and Initialize the ZMF variable pool dataset.
 /*
 DEFINE CLUSTER (NAME(CMNDEV.CMNSYS.U810ALL.CMNVPOOL)          +
   RECORDSIZE(40,4096) UNIQUE INDEXED KEYS(24,0)               +
   FREESPACE(20 20)                                            +
   SHAREOPTIONS(2,3))                                          +
  DATA(NAME(CMNDEV.CMNSYS.U810ALL.CMNVPOOL.DATA)               +
   CYLINDERS(1, 1)                                            +
   CISZ(32768))                                               +
  INDEX(NAME(CMNDEV.CMNSYS.U810ALL.CMNVPOOL.INDEX)             +
   CYLINDERS(1,1)                                             +
   CISZ(2048))

 REPRO IFILE(DUMMY)    ODS(CMNDEV.CMNSYS.U810ALL.CMNVPOOL)

//*
//* Delete DUMMY record from VSAM file
//*
//VPOOL   EXEC PGM=CMNVINIT
//CMNVSAM   DD DISP=SHR,DSN=CMNDEV.CMNSYS.U810ALL.CMNVPOOL
```

The key to the variable value is an 8-byte pool name concatenated with a 16-byte variable name. The length of the variable data is up to 4070 bytes.

You can choose any name for the pool name but it will usually be the userid so that any single user can own his or her own copy of a set of common variables (much like a set of ISPF profile variables). However, obviously, global or application (or any other) groupings of variables can be implemented, completely at the customer's discretion.

The actual variable name is limited to 16 bytes in length and is fixed in length (but can be padded with blanks as necessary). We then have the variable value which is stored in a VARCHAR format. An example of one such variable residing in our test data set looks like this:

```
....+....1....+....2....+....3....+....4..
WSER58  TrialVariable   ..CheckThisOut
EECDFF44E9889E89888898444401C8889E88ADAA4444
625958003991351991235000003853238926430000
```

We have a pool name of WSER58 (padded to 8 bytes) and a variable name of TrialVariable (padded to 16 bytes). This is followed by the length of the value (highlighted in red), x'0010' = 16, in this case. The value of the variable follows.

In the HLLX procedure JCL this KSDS is defined thus:

```
//*
//CMNVPALT  DD DISP=SHR,DSN=CMNDEV.CMNSYS.U810ALL.CMNVPOOL
//CMNVPOOL  DD SUBSYS=(BLSR,'DDNAME=CMNVPALT','STRNO=255')
//*
```

The functions provided to allow access to these variables are primarily intended for use within the HLLX address space. However, they have also been coded so that they can be used anywhere that the VSAM KSDS has been allocated to ddname CMNVPOOL. However, note that functions executing outside of the HLLX environment are only allowed read access to the CMNVPOOL VSAM file, thereby limiting the available functions to INIT, GET, and TERM.

The HLLX address space will open the CMNVPOOL ACB (if the ddname is present) as part of its initialization process, and will close it on termination. If there is a problem opening the CMNVPOOL file, the following message will be issued:

```
07.11.16 S0069082  CMNX100E Failed to open CMNVPOOL KSDS, fdbk: 000000BC
```

The fdbk code is a standard VSAM open macro feedback code, which, in this case, is indicating that the data set we are attempting to open is not a VSAM file.

If any of the vpool functions are running under HLLX (the code detects this) they assume the ACB is already available in HLLX common storage. Each user subtask works with its own RPL storage. If the functions are invoked outside of the HLLX address space, the ACB is opened as part of the process. (That is, the functions will work inside or outside the HLLX address space but will be far more efficient within.)

The calls supported are:

Call	Description
INIT:	Establishes a conversation within which the other functions may be executed without incurring repeated initialization overhead. This is not so important within the HLLX address space as the biggest overhead is in setting up and opening the ACB (which is already done by the HLLX main task). Any conversation begun with an INIT call must be ended with a TERM call. If INIT/TERM are not used then each individual call is wrapped with an internal INIT/TERM pair.
TERM:	Terminates a conversation previously started via an INIT call.
DEF:	Defines a variable (similar to an ISPF VDEFINE call). It is used, in particular, to establish a length for the variable value so that subsequent processes have a solid reference when moving data around (thereby avoiding S0C4s). The call writes a record to the VSAM file of the appropriate length and filled with blanks. If the record already exists then we return the length already defined for it and RC=4. Only available from within HLLX.
DEL:	Deletes the variable record. (To change the length of a previously defined variable you would first delete it and then define it again with the new length.) Only available from within HLLX.
PUT:	Updates the variable record with a value. Only available from within HLLX.
GET:	Extracts the current variable value.

CMNLPOOL is intended to be called from LE programs using a standard call parameter list:

```
 ***************************************
 *DEFINE THE PARAMETER LIST USED BY THE
 *VARIABLE POOL ROUTINE  - CMNPOOL
 *Function may be one of:
 *   INIT  - INITALISE THE VARIABLE POOL
 *   DEF   - DEFINE A VARIABLE
 *   DEL   - DELETE A VARIABLE
 *   PUT   - DEFEINE THE VALUE OF THE VARIABLE
 *   GET   - RETRIEVE THE VALUE OF A VARIABLE
 *   TERM  - TERMINATE THE VARIABLE POOL
 ***************************************
 *
      03  WS-VP-FUNCTION         PIC X(4)   VALUE SPACES.
      03  WS-VP-MSGAREA          PIC X(128) VALUE ' '.
      03  WS-VP-CONTEXT          PIC S9(8)  COMP VALUE +0.
      03  WS-VP-POOL             PIC X(8)   VALUE SPACES.
      03  WS-VP-VARNAME          PIC X(16)  VALUE SPACES.
      03  WS-VP-VARLEN           PIC S9(4)  VALUE +256.
      03  WS-VP-VARVALUE         PIC X(256) VALUE SPACES.
 *
```

A sample call sequence is shown below. Note that the INIT/TERM calls are not strictly necessary within HLLX but are recommended for use outside of HLLX:

```
*******************
*            *1)Start a VPOOL conversation*
*            *2)Define a variable         *
*            *3)Modify that variable      *
*            *4)Get its updated value     *
*            *4)End the VPOOL conversation*
*            ****************************
*
*            ****************************
*            * Initialise the variable    *
*            * pool access conversation   *
*            ****************************
     MOVE 'INIT' TO WS-VP-FUNCTION.
     CALL CMNLPOOL USING WS-VP-FUNCTION
                         WS-VP-MSGAREA
                         WS-VP-CONTEXT.
     DISPLAY 'INIT RETURN-CODE: ' RETURN-CODE.
     DISPLAY 'INIT CONTEXT    : ' WS-VP-CONTEXT.
*
*            ****************************
*            * Define a variable - TestVar*
*            * data length - 256          *
*            * assigned to pool - WSER58  *
*            ****************************
     MOVE 'DEF ' TO WS-VP-FUNCTION.
     MOVE 'TestVar' TO WS-VP-VARNAME
     MOVE 256 TO WS-VP-VARLEN.
     MOVE 'WSER58' TO WS-VP-POOL.
*
     CALL CMNLPOOL USING WS-VP-FUNCTION
                         WS-VP-MSGAREA
                         WS-VP-CONTEXT
```

```
                        WS-VP-POOL
                        WS-VP-VARNAME
                        WS-VP-VARLEN.
     DISPLAY 'DEF  RETURN-CODE: ' RETURN-CODE.
     DISPLAY 'DEF  VARLEN     : ' WS-VP-VARLEN.
*
*            *************************
*            * Assign a value for the *
*            * variable just defined  *
*            *************************
     MOVE 'PUT ' TO WS-VP-FUNCTION.
     MOVE 'BLAH' TO WS-VP-VARVALUE.
*
     CALL CMNLPOOL USING WS-VP-FUNCTION
                         WS-VP-MSGAREA
                         WS-VP-CONTEXT
                         WS-VP-POOL
                         WS-VP-VARNAME
                         WS-VP-VARVALUE.
     DISPLAY 'PUT  RETURN-CODE: ' RETURN-CODE.
*
*            ***********************************
*            * Retrieve the value of a variable *
*            ***********************************
     MOVE 'GET ' TO WS-VP-FUNCTION.
     MOVE SPACES TO WS-VP-VARVALUE.
*
     CALL CMNLPOOL USING WS-VP-FUNCTION
                         WS-VP-MSGAREA
                         WS-VP-CONTEXT
                         WS-VP-POOL
                         WS-VP-VARNAME
                         WS-VP-VARVALUE.
     DISPLAY 'GET  RETURN-CODE: ' RETURN-CODE.
     DISPLAY 'GET  VARVALUE   : ' WS-VP-VARVALUE.
*
*            ****************************
*            * Terminate variable pool    *
*            * access conversation        *
*            ****************************
     MOVE 'TERM' TO WS-VP-FUNCTION.
*
     CALL CMNLPOOL USING WS-VP-FUNCTION
                         WS-VP-MSGAREA
                         WS-VP-CONTEXT.
     DISPLAY 'TERM RETURN-CODE: ' RETURN-CODE.
*
```

REXX execs need to call CMNRPOOL using the LINKMVS command. (CMNRPOOL relies on the parameter structure generated by LINKMVS.) Again, INIT/TERM are not strictly

necessary within HLLX but are shown here for completeness. There are a number of special REXX variables returned by CMNRPOOL:

Variable	Description
VPOOLMSG	Contains any messages returned by the process. (Note: The standard REXX variable RC contains the return code.)
VPOOLVLN	The length of the variable just defined.
VPOOLCTX	Hexadecimal value representing the conversation context, generated by an INIT call. It is passed from one execution of CMNRPOOL to the next until the TERM call is made. It is purely for internal use and no good will come from tampering with it.

```
/*  Demonstration of the use of the HLLX vpool facility   */
/*  REXX execs call the CMNRPOOL front end program        */
/*  LE programs use the CMNLPOOL front end program        */
/*  However, after differing parameter parsing, both pass */
/*  control to the same CMNVPOOL subroutine.              */
/*                                                        */
/*  Any messages are returned in REXX variable VPOOLMSG.  */
/*  The length of a defined variable is returned in       */
/*  VPOOLVLN.                                             */
/*  If conversational mode is setup then the context is   */
/*  held in VPOOLCTX, but this is for internal use only.  */

Say " "
Say "Demonstration of HLLX variable pool services"
Say " "

function = "INIT"
address LINKMVS "CMNRPOOL function"
Say "Return Code from INIT call is: "RC
Say "Returned message is          : "VPOOLMSG
Say " "

function = "DEF"
userid   = "WSER58"
varname  = "TrialVariable"
varlen   = "16"
address LINKMVS "CMNRPOOL function userid varname varlen"
Say "Return Code from DEF  call is: "RC
Say "Returned message is          : "VPOOLMSG
Say "Variable length is           : "VPOOLVLN
Say " "

TrialVariable = "CheckThisOut"

function = "PUT"
address LINKMVS "CMNRPOOL function userid varname"
Say "Return Code from PUT  call is: "RC
Say "Returned message is          : "VPOOLMSG
Say "TrialVariable after PUT      : "TrialVariable
Say " "
```

```
TrialVariable = "ChangeIt"

Say "TrialVariable before GET     : "TrialVariable

function = "GET"
userid   = "WSER58"
varname  = "TrialVariable"

address LINKMVS "CMNRPOOL function userid varname"
Say "Return Code from GET  call is: "RC
Say "Returned message is          : "VPOOLMSG
Say "TrialVariable after GET      : "TrialVariable
Say " "

function = "TERM"
address LINKMVS "CMNRPOOL function"
Say "Return Code from TERM call is: "RC
Say "Returned message is          : "VPOOLMSG
Say " "
```

Tracing

The HLLEXIT service requests are just like any other service request and, as such, can be traced using the standard CMN/Sernet tracing facilities. This trace will show you what the client is sending to the ZMF started task.

At the other end of the HLLX path, each exit can also use display/put/say facilities to show what it is being passed.

The data is reformatted by the code executing in the HLLX started task. There is a new trace facility available to show exactly what has been passed to the HLLX started task prior to this reformatting. (Note: Displays in the exit can show what the reformatted data areas look like as they are passed directly to the exit after reformatting). The trace will also show what has been passed back to the HLLX started task from the user exit after any reformatting.

Although this tracing is happening in the HLLX address space, it is controlled with modify commands to the ZMF/Sernet started task (to avoid having to work with more than one started task). The trace command is modeled after the NETTRACE Sernet command, for example:

```
/F zmfstcname,HLXTRACE,ON,EXIT=exitmask,USER=usermask
```

OFF turns off all traces.

There is no sophistication to this trace command. OFF turns off the prior defined trace request. Each new ON command replaces the prior trace definition.

As with NETTRACE there are shortcut synonyms for HLXTRACE (HT) and ON/OFF (Y/N).

The mask fields work with an asterisk at the end of the value only. For example:

```
/F stcname,HT,Y,EXIT=PCRE*,USER=WSER5*
```

will put out trace information for all PCRE*xxxx* user exit points and for all userids beginning with WSER5. As usual with trace commands, the more specific you can be the less trace output that you will have to examine.

The length of the data displayed is taken from the length field at the beginning of each data area. If this is found to be non-numeric then a default of 256 bytes is employed.

Here is some sample trace output:

```
20141027 01:52:34.91  USERID=WSER58       Connect for function: PCRE, Token: 00000000000000030000017800000002

20141027 01:52:35.54  USERID=WSER58     Input data for exit: PCRE0001
 0000: F0F1F7F2 F2F0F0F0 F0F0F0F0 F0F0F0F0   F0F0F0F0 F3F0F0F0 F0F0F1F7 F8F0F0F0   *0172200000000000000003000001780000*
 0020: F0F0F0F0 F2D7C3D9 C5F0F0F0 F1E8E2D7   C6D9C4F1 F0C7E6E2 C5D9F5F8 4040E3E2   *00002PCRE0001YSPFRD10GWSER58  TS*
 0040: E37BD7C3 D9C54040 40404040 40404040   40404040 40404040 40404040 40404040   *T#PCRE                          *
 0060: 40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
 0080: 40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
 00A0: 40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
 00C0: 40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
 00E0: 40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
 0100: 40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
 0120: 40404040 40404040 40404040 40404040   40404040 40404040 40404040 4040E2E3   *                              ST*
 0140: C5E5F240 40404040 40404040 40404040   40404040 404040F1 F1404040 40404040   *EV2                     11       *
 0160: 40404040 4040C4C5 D7E3E2A3 85A58540   C496A695 85A24040 40404040 40404040   *      DEPTSteve Downes          *
 0180: 404040F9 F1F14040 40404040 40404040   4040F140 40404040 40404040 40404040   *   911            1             *
 01A0: 40404040 40404040 40404040 40404040   40404040 40404040 40404040 404040F1   *                               1*
 01C0: 404040F1 F2F3F4F5 F6F7F8F9 404040E3   85A2A340 D7818392 81878540 40404040   *   123456789   Test Package     *
 01E0: 40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
 0200: 40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   *                                *
...
```

Chapter 3

ZMF/HLL Exit Interface

Introduction

Information is passed from the client to the HLL exit address space for processing and sending on to the user exit itself. If the exit is designated as a REXX exit, REXX variables are defined and populated with values from the incoming request.

If the exit is an LE-program, the incoming data is formatted in such a way that a (supplied) COBOL and/or PL/I copybook can be used to map the data.

On entry to an exit, a field (*xxxx*ORGN for LE code and callOrigin as a REXX variable) will be set to an identifier that identifies the client process that resulted in this call. These identifiers are:

Identifier	Description
ECL	The call is as a result of a ZMF4ECL client process.
ZDD	The call is as a result of a ZDD client process.
SPF	The call is as a result of an ISPF client process.
XML	The call is due to a direct invocation of an XML service (for example, Through XMLSERV or equivalent).

The exit has the opportunity to update that data in place. If it does so, it must set the dataChanged variable (or LE equivalent) to YES; otherwise ZMF will ignore it.

If an exit wishes to stop a process, it can set the proceed variable to NO. It can also populate the shortMsg and longMsg variables to whatever is required to be displayed for the condition leading to the stopping of the current function. Furthermore, it can set the cursorField variable to position the cursor at a specific field.

All normal execution of HLL exit routines must end with RC=0 (note that EXIT with no expression is treated as EXIT 0 by HLLX). If the routine ends with RC>0, the infrastructure will take this as a major failure and abandon the current function altogether. In this case, the client will produce a general message indicating an HLL exit failure and will direct the user to the HLLX started task output for further details. (A developer should contact his or her administrator at this point.)

For an LE program we see something like this in sysout:

```
LE program for exit function PCRE0007 finished with RC=00000007
```

And for a REXX exec we have something like:

```
REXX RC for exit function PCRE0101 is 00000000
REXX evaldata (expression coded on EXIT statement) is 9
```

These messages will mean something to the exit developer as they will have set the non-zero return code. Note that the return code from a REXX exec (for example, EXIT *nn*) is returned in the evalblock (as shown above for EXIT 9). The actual return code from the invocation of IRXEXEC in this case is zero (that is, successful execution of the exec). The return code will be non-zero only for situations such as REXX syntax errors which cause the execution to fail.

On return, the HLL exit address space extracts and populates the data in the response section of the XML service request, which gets passed back to the client to deal with as it wishes.

Build

This section describes the build functional area of the high-level language exits. The build function includes component checkin, stage, build, recompile, and relink.

The 4-character exit name identifier is BULD.

Select option 2 Build from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define customized ISPF variables for the build function:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed.

The panels around which exit points will be placed are listed below. The internal exit name (also known as function code) is BULD0*pnn*, where:

- *p*=0 is the pre-exit.
- *p*=1 is the post-exit.
- *nn* is an alphanumeric identifier relating to the panel for which the exit is taken.

An internal exit name of BULD0*p*01, for example, means that both pre- and post-exits exist. That is, the name of the pre-exit is BULD0001 and the name of the post-exit is BULD0101. The pre-exit is taken before the panel is displayed and the post-exit is taken after the panel has been displayed.

Many panels in these dialogs are either menu driven or consist of selection lists from which actions are performed against selected entries. Owing to the potentially huge lists that would have to be built and passed to pre-exits for these panels, and also the potential to adversely affect ZMF processing if the user exit is coded incorrectly, these panels will only have post-exits taken once for each selected entry. The post-exits can be used to validate selected entries as required. Such panels are indicated in the following lists by means of an asterisk.

Stage:

Panel ID	Description	Exit Name
CMNSTG00*	Main stage function menu	BULD0100
CMNSTG01/14/24*	Package component list (short/long/xlong	BULD0101
CMNSTG02	Stage from development initial	BULD0002 BULD0102
CMNSTG03*	Stage from development member selection	BULD0103
CMNSTG04	Standard stage job submission	BULD0004 BULD0104
CMNSTG05	Mass stage job submission	BULD0005 BULD0105
CMNSTG06*	Procedure selection	BULD0106
CMNSTG07*	Language selection	BULD0107
CMNSTG08*	Libtype selection	BULD0108
CMNSTG09	Like-Other standard stage job submission	BULD0009 BULD0109
CMNSTG10	Like-Other mass stage job submission	BULD0010 BULD0110
CMNSTG11*	RECFM=U stage from dev member selection	BULD0111
CMNSTG12	Component selection parameters	BULD0012 BULD0112
CMNSTG15	Component general description	BULD0015 BULD0115
CMNSTG17*	Valid staging line commands	BULD0117
CMNSTG18*	Db2 subsystem selection	BULD0118
CMNSTG19	Batch staging job card definition	BULD0019 BULD0119
CMNSTG20	Confirm delete request	BULD01DL
CMNSTG23*	Stage from development (zFS) file selection	BULD0123
CMNUSR01-04	Like-SRC component user variables	BULD00US BULD01US
CMNUSR11-13	Non-SRC component user variables	BULD00US BULD01US

Recompile:

Panel ID	Description	Exit Name
CMNRCMP0 CMNRCMPR	Main recompile entry	BULD00R0 BULD01R0
CMNRCMP1	Standard recompile submission	BULD00R1 BULD01R1

Panel ID	Description	Exit Name
CMNRCMP2/4/5/6/7*	Component list (various formats)	BULD01R2
CMNRCMP3	Mass/batch recompile submission	BULD00R3 BULD01R3
CMNUSR01-04	Like-SRC component user variables	BULD00US BULD01US
CMNRCMPC*	Recompile confirmation	BULD01RC
CMNSTG06*	Procedure selection	BULD0106
CMNSTG07*	Language selection	BULD0107
CMNSTG08*	Libtype selection	BULD0108
CMNSTG18*	Db2 subsystem selection	BULD0118
CMNLBLST CMNLBLSR	Recompile from library selection	BULD01RL
CMNLBLS2*	Promotion library selection	BULD01RP
CMNLBLS3*	Release library selection	BULD01RR

Relink:

Panel ID	Description	Exit Name
CMNRLNK0 CMNRLNKR	Main relink entry	BULD00L0 BULD01L0
CMNRLNK1	Relink job submission	BULD00L1 BULD01L1
CMNRLRLS*	Relink from library list vis release search	BULD01LR
CMNRMLST/T2/OD/D2*	Member list (various formats)	BULD01LT
CMNSTG06*	Procedure selection	BULD0106
CMNSTG07*	Language selection	BULD0107
CMNSTG18*	Db2 subsystem selection	BULD0118
CMNRLTYP	Output library type selection	BULD01LO

XML build services:

XML Service Name	Description	Exit Name
cmponent.checkin.service	Check in	BULD00XC BULD01XC
cmponent.build.service	Build	BULD00XB BULD01XB
cmponent.recomp.service	Recompile	BULD00XR BULD01XR
cmponent.relink.service	Relink	BULD00XL BULD01XL

Note that no data values are taken from exits BULD01XC/B/R/L (that is, after the XML request has completed) as no further processing takes place in the function to which changes in data values can be passed. These exits are purely to allow customers to initiate whatever post-build function external processes they desire.

If BULDLOKD is set to YES, data fields on the related panel will be set to output only. The list of panels for which this applies is:

Stage:

Panel ID	Description	Exit Name
CMNSTG04	Standard stage job submission	BULD0004
CMNSTG05	Mass stage job submission	BULD0005
CMNSTG09	Like-Other standard stage job submission	BULD0009
CMNSTG10	Like-Other mass stage job submission	BULD0010
CMNSTG15	Component general description	BULD0015
CMNSTG19	Batch staging job card definition	BULD0019
CMNUSR01-04	Like-SRC component user variables	BULD00US
CMNUSR11-13	Non-SRC component user variables	BULD00US

Recompile:

Panel ID	Description	Exit Name
CMNRCMP1	Standard recompile submission	BULD00R1
CMNRCMP3	Mass/batch recompile submission	BULD00R3

Relink:

Panel ID	Description	Exit Name
CMNRLNK1	Relink job submission	BULD00L1

BULDSHRT and BULDLONG are used to set a message on the next panel/window to be displayed.

If BULDGO is set to NO, the fields BULDSHRT, BULDLONG, and BULDCURS will be used to set an error message, and the client will (re)display the associated panel.

If the user exit wishes to change any of the data fields, it does that in place and sets BULDCHNG to YES. If BULDCHNG is not set to YES, the client ignores any data changes.

The various build actions (e.g. stage, recompile, relink etc.) take different exits which may have different sets of input and/or modifiable fields (with most in common). Which exits/fields are available should be obvious from the context. Some of the query functions take only exits BULD0xUS and the fields modifiable by these exits are indicated by notes 4,5 & 6.

A single data structure is passed to all of these exits. The data interface for the build exits looks like this:

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
BULDFUNC	function	8	Function code	No	
BULDDBUG	debugCall	1	Debug exit call (Y/N)?	No	
BULDORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL	No	
BULDZMFS	zmfSubs	1	ZMF subsystem Id	No	
BULDPDB2	db2Subs	4	Primary Db2 subsystem	No	
BULDUSER	userid	8	Userid	No	
BULDEXTN	externalName	256	External name for exit	No	
BULDLOKD	dataLocked	3	Fields locked (YES/NO)?	Yes	
BULDGO	proceed	3	Proceed (YES/NO)?	Yes	
BULDSHRT	shortMsg	24	Short message	Yes	
BULDLONG	longMsg	128	Long message	Yes	
BULDCURS	cursorField	3	Cursor tag	Yes	
BULDCHNG	dataChanged	3	Data changed (YES/NO)?	Yes	
BULDPKGN	packageId	10	Package name	No	
BULDPSTA	packageStatus	3	Package status (for example, DEV)	No	
BULDPINS	packageInsDate	8	Package install date	No	
BULDCOMP	component	256	Component name	No	001
BULDLTYP	componentType	3	Component libtype	Yes (note #4,5,6)	002
BULDCSTA	componentStatus	8	Component status	No	003
BULDSDTE	stageDate	8	Component stage date	No	004
BULDSTME	stageTime	6	Component stage time	No	005
BULDFTYP	stageFunction	6	Function type	No	
BULDLCMD	lineCommand	4	Line command entered	No	006
BULDPROC	buildProc	8	Compile procedure	Yes (note #4)	007
BULDLANG	language	8	Language	Yes (note #4)	008
BULDCOPT	compileOptions	34	Compile options	Yes (note #4)	009

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
BULDLOPT	linkOptions	34	Program binder options	Yes (note #4)	010
BULDMODE	buildMode	1	Stage mode	Yes	011
BULDCUSR	componentUserOptions	1	Component user variables	Yes	012
BULDEUSR	extUserOptions	1	Extended user variables	Yes (note #4)	013
BULDLOCK	lockComponent	1	Lock component	Yes	014
BULDCNFM	confirmAction	1	Confirm actions	Yes	015
BULDHFSX	expandHfsDirs	1	Expand HFS directories	Yes	016
BULDPCVR	showDb2Panels	1	Db2 precompile information	Yes	017
BULDSUPN	suppressNotify	1	Suppress notify messages	Yes	018
BULDPSIT	recompileSite	8	Recompile site	Yes	019
BULDBLVL	recompileLevel	3	Recompile level	Yes	020
BULDSRLS	searchOrSelectRelease	1	Search/specify release	Yes	021
BULDSARE	recompileSelectArea	1	Recompile select area	Yes	022
BULDRLSE	recompileRelease	8	Recompile release	Yes	023
BULDAREA	recompileArea	8	Recompile area	Yes	024
BULDLLVL	relinkFromSBR	1	Relink from S/B/R	Yes	025
BULDFLCT	relinkUsingLCT	1	Relink using LCT	Yes	026
BULDUHST	useHistory	1	Use history for prms	Yes	027
BULDIJNI	incrementJobname	1	Job name increment	Yes	028
BULDDB2P	useDb2PreCompileOption	1	Db2 precompile (Y/N)?	Yes (note#4)	029
BULDDB2R	db2RemoteSite	8	Db2 remote site	Yes	141
BULDDB2S	db2SubSystemId	4	Db2 subsystem id	Yes	030
BULDSPLC	db2SpLocation	16	Db2 SP location	Yes	140
BULDDB2L	db2PreCompileLinkLib	44	Db2 library name	Yes	031
BULDDPCV	db2PreCompileVersion	64	Db2 precompiler version	Yes	032
BULDJOB1	jobCard01	72	Job card line 1	Yes	033
BULDJOB2	jobCard02	72	Job card line 2	Yes	034
BULDJOB3	jobCard03	72	Job card line 3	Yes	035
BULDJOB4	jobCard04	72	Job card line 4	Yes	036
BULDODSN	inputDataset	44	Like other input data set name	No	138

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
BULDSLOC	sourceLocation	1	Input source location	Yes	
BULDPRJ0	prj0	8	Input ISPF library high-level qualifier	Yes (note#1)	037
BULDLIB0	lib0	32	Input ISPF library mid-level qualifier	Yes (note#1)	038
BULDTYP0	typ0	8	Input ISPF library low-level qualifier	Yes (note#1)	039
BULDIMBR	inputMember	8	Input ISPF library member	Yes (note#1)	040
BULDIORG	inputDSorg	3	Input library DSORG	No	041
BULDVVMM	inputMemberVvMm	5	Input member *vv.mm*	No	042
BULDCRTD	inputMemberCreateDate	10	Input member create date	No	043
BULDCHGD	inputMemberChangedDate	10	Input member change date	No	044
BULDCHGT	recompileChangeTime	5	Recompile change time	No	045
BULDCSZE	inputMemberCurrentSize	5	Input member current size	No	046
BULDISZE	inputMemberInitialSize	5	Input member initial size	No	047
BULDCUID	inputMemberChangeUserid	8	Input member change userid	No	048
BULDLSZE	loadMemberSize	6	Load member size	No	049
BULDTTR	loadMemberTTR	6	Load member TTR	No	050
BULDALIS	loadMemberAliasOf	8	Load member alias of	No	051
BULDAUTH	loadMemberAuthCode	2	Load member authorization code	No	052
BULDRMOD	loadMemberRmode	3	Load member Rmode	No	053
BULDAMOD	loadMemberAmode	3	Load member Amode	No	054
BULDSSSI	loadMemberSetSSI	8	Load member SETSSI	No	055
BULDATTR	loadMemberAttributes	15	Load member attributes	No	056
BULDDMOD	componentListMode	1	Component list display	Yes (note#2)	057
BULDCNFD	confirmDelete	1	Confirm delete (Y/N)?	Yes (note#2)	058
BULDCMPR	comparisonReport	1	Comparison report (Y/N)?	Yes (note#2)	059
BULDCMPT	comparisonText	8	Comparison text	Yes (note#2)	060
BULDTLTP	targetLibtype	3	Target libtype	Yes	061

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
BULDUPAN	userPanel	8	User variable panel	Yes (note#4,5,6)	
BULDUVAR	userVariables	1	Display User variable panel (Y/N)	Yes	
BULDOPRF	optsProfile	8	Used to select the ZDDOPTS profile for the display of user options for the ZMF Client Pack	Yes	
Component User Variables					
BULDUO1	userOptionsPart1	1 * 10	Set of ten 1-byte user options (user options 01-10)	Yes (note#4)	062-071
	userOption01-10	1	REXX variables which map each individual byte of userOptionsPart1	Yes (note#4)	062-071
BULDUO2	userOptionsPart2	1 * 10	Set of ten 1-byte user options (user options 11-20)	Yes (note#4)	072-081
	userOption011-20	1	User variable panel name	Yes (note#4)	072-081
BULDUVPN	userVarPanel	8	Set of five 8-byte user variables (user variables 1-5)	Yes	
BULDUV1	userVariable01-05	8 * 5	Set of five 8-byte user variables (user variables 1-5	Yes	082-086
BULDUV6	userVariable06-10	72 * 5	Set of five 72-byte user variables (user variables 6-10)	Yes	087-091
BULD01	userOption0101-0105	1 * 5	Set of five 1-byte user variables (user variables 0101-0105)	Yes (note#4)	092-096
BULD02	userOption0201-0203	2 * 3	Set of three 2-byte user variables (user variables 0201-0203)	Yes (note#4)	097-099
BULD03	userOption0301-0303	3 * 3	Set of three 3-byte user variables (user variables 0301-0303)	Yes (note#4)	100-102
BULD04	userOption0401-0403	4 * 3	Set of three 4-byte user variables (user variables 0401-0403)	Yes (note#4)	103-105
BULD08	userOption0801-0805	8 * 5	Set of five 8-byte user variables (user variables 0801-0805)	Yes (note#4)	106-110

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
BULD10	userOption1001-1002	10 * 2	Set of two 10-byte user variables (user variables 1001-1002)	Yes (note#4)	111-112
BULD16	userOption1601-1602	16 * 2	Set of two 16-byte user variables (user variables 1601-1602)	Yes (note#4)	113-114
BULD34	userOption3401-3402	34 * 2	Set of two 34-byte user variables (user variables 3401-3402)	Yes (note#4)	115-116
BULD44	userOption4401-4402	44 * 2	Set of two 44-byte user variables (user variables 4401-4402)	Yes (note#4)	117-118
BULD64	userOption6401-6405	64 * 5	Set of five 64-byte user variables (user variables 6401-6405)	Yes (note#4)	119-123
BULD72	userOption7201-7205	72 * 5	Set of five 72-byte user variables (user variables 7201-7205)	Yes (note#4)	124-128
Db2 Information					
BULDXDB2	extractFromDb2	1	Extract from Db2	Yes	129
BULDXD2I	extractFromDb2Id	4	Extract from Db2 id	Yes	130
BULDSPSC	db2SpSchema	128	Db2 stored procedure schema	Yes	131
BULDSPNM	db2SpName	8	Db2 stored procedure name	Yes	132
BULDSPVR	db2SpVersion	122	Db2 stored procedure version	Yes	133
BULDSPVI	db2SpVersionInd	1	Db2 stored procedure version ind	Yes	134
BULDSPZI	db2SpZmfInfo	1	Db2 stored procedure add ZMF information	Yes	135
VARCHAR Area for Data Set Name					
BULDDSNM-LEN		2	Data set name length	Yes	
BULDDSNM-VALUE	fileOrDsname	1280	Data set name value	Yes (note#1)	
Component Description					
BULDCDSC	componentDesc.n	72 * 48	Up to forty-eight 72-byte lines	Yes (note#3)	

Note#1: Only modified from exit BULD0x02

Note#2: Only modified from exit BULD0x12

Note#3: Only modified from exit BULD0x15

Note#4: Additionally modifiable in the component admin functions where BULD0xUS exits are taken.

Note#5: Additionally modifiable in the component display functions where BULD0xUS exits are taken.

Note#6: Additionally modifiable in the general pmast/cmast query functions where BULD0xUS exits are taken.

Package Create

This section describes the package-create functional area of the high-level language exits. The 4-character exit name identifier is PCRT.

Select option 3 Package Create from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define customized ISPF variables for the package-create function:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed.

Panels around which exit points will be placed are listed below. The internal exit name (also known as function code) is PCRE0*pnn*, where:

- *p*=0 is the pre-exit.
- *p*=1 is the post-exit.
- *nn* is an alphanumeric identifier relating to the panel for which the exit is taken.

The pre-exit is taken before the panel is displayed (usually to provide model parameters displayed on the panel); the post-exit is taken after the panel has been displayed (for example, for input validation or enforcement).

For the package-create function, we are building the complete set of information as we progress through the dialog. The exit data format is constant throughout the process but fields may not be filled in depending on where the exit is in the process.

Package Create:

Panel ID	Panel Description	Exit Name
CMNCRT01	Initial package-create panel (non-ERO)	PCRE0p01
CMNCRT0R	Initial package-create panel (ERO)	PCRE0p01
CMNCRT02	Package description (long form create only)	PCRE0p02
CMNCRT03	Installation instructions (long form create only)	PCRE0p03
CMNCRT04	Scheduling dependencies	PCRE0p04
CMNCRT05	Affected applications	PCRE0p05

Panel ID	Panel Description	Exit Name
CMNDPUP1 CMNDPUP2	Package user options	PCRE0pPU
CMNCRT06	Install time/date, etc. (ALL site)	PCRE0p06
CMNCRT07	Site list with install date/time, etc. (DP site)	PCRE0p07
CMNCRT08	Complex/Super information	PCRE0p08

XML Package-Create Service:

Service	Exit Name
package.create.service	PCRE0pXM

Note that no data values are taken from exit PCRE01XM (that is, after the XML request has completed) as no further processing takes place in the function to which changes in data values can be passed. The PCRE01XM exit is purely to allow you to initiate whatever post-package-create external processes you desire.

The basic framework for the data structure passed to all exits is the set of fields accepted by the package create service. In addition, we will pass fields that allow you to make further decisions and pass back information such as error messages and so on.

The XML package-create service pre-exit will have the exit call data formatted from the input service tags and will be taken before any of the service processing. The post-exit is taken after the package has been created and is the only exit to which the created package name is passed (all other exits being taken before the package has been created). You cannot change anything or affect processing from the XML service post-exit: It is intended to be used as a mechanism of notifying external processes that the package has been created.

In order to allow the cursor to be positioned at specific panel fields, a number relating to the panel field in question will be passed back in the XPCROCURS field. These numbers are documented in the supplied copybook for the function (and, eventually, in the lists of REXX field names). It is your responsibility to make sure that your exit returns a field number that is valid for the panel about to be (re)displayed. If the field number is not valid, a dialog error results.

The package-create format shown below (in COBOL v5.1 format for compactness) is for illustration purposes. Copybooks will be created for COBOL (compatible with earlier versions of COBOL) and PL/I. The equivalent REXX variable list will also be published.

The request block of each HLL exit service request starts off with a few general fields followed by package-create function-specific fields.

A COBOL example:

```
 01  PCRT.                              *>REQUEST ELEMENT LAYOUT
***
* PACKAGE CREATE PROCESS HLL EXITS - PASSED VARIABLES
***
     03 PCRTFUNC            PIC X(8).  *>FUNCTION CODE
     03 PCRTDBUG            PIC X(1).  *>DEBUG EXIT CALL? Y/N
     03 PCRTORGN            PIC X(3).  *>CALL ORIGIN
     03 PCRTZMFS            PIC X(1).  *>ZMF SUBSYSTEM ID
     03 PCRTPDB2            PIC X(4).  *>PRIMARY Db2 SUBSYSTEM
     03 PCRTUSER            PIC X(8).  *>USER ID
     03 PCRTEXTN            PIC X(156). *>EXTERNAL EXIT NAME
     03 PCRTAPPL            PIC X(4).  *>APPLICATION
```

```
     03 PCRTMETH            PIC X(1).  *>PKG CREATE METHOD
     03 PCRTLOKD            PIC X(3).  *>FLDS LOCKED? YES/NO
     03 PCRTGO              PIC X(3).  *>PROCEED? YES/NO
     03 PCRTSHRT            PIC X(24). *>SHORT MESSAGE
     03 PCRTLONG            PIC X(128). *>LONG MESSAGE
     03 PCTRCURS            PIC X(24). *>CURSOR TAG
     03 PCRTCHNG            PIC X(3).  *>DATA CHANGED? YES/NO
     03 PCRTCFPK            PIC X(10). *>NAME OF PKG TO CARRY FWD
     (More fields here. (See "Package Create" on page 57 a for full listing.)
     03 PCRTRARE            PIC X(8).  *>RELEASE AREA
***
* VARIABLE BLOCK POINTERS
***
     03 PCRTVB1L-PTR        USAGE IS POINTER.
     03 PCRTVB2L-PTR        USAGE IS POINTER.
     03 PCRTVB3L-PTR        USAGE IS POINTER.
     03 PCRTVB6L-PTR        USAGE IS POINTER.
     03 PCRTVB7L-PTR        USAGE IS POINTER.
     03 PCRTVBSL-PTR        USAGE IS POINTER.
***
* PACKAGE DESCRIPTION - UP TO 46 LINES OF 72 BYTES
***
 01  PCRTVB1.
     03 PCRTPDSC            PIC X(72). *>LINE OF DESCRIPTION
     03  PTR-NEXT-PCRTVB1   POINTER.   *>POINTER TO NEXT LINE
***
* IMPLEMENTATION INSTRUCTIONS - UP TO 46 LINES OF 72 BYTES
***
 01  PCRTVB2.
     03 PCRTPIMI            PIC X(72). *>LINE OF INSTRUCTION
     03  PTR-NEXT-PCRTVB2   POINTER.   *>POINTER TO NEXT LINE
***
* SCHEDULING INFORMATION - LIMITED ONLY BY STORAGE CONSTRAINTS
***
 01  PCRTVB3.
     03 PCRTSCHI.
        05 PCRTSSSJ         PIC X(8).  *>SUCCESSOR JOBNAME
        05 PCRTSSPJ         PIC X(8).  *>PREDECESSOR JOBNAME
     03  PTR-NEXT-PCRTVB3   POINTER.   *>POINTER TO NEXT SET
...
```

The user exit should follow the pointer chains for each repeating group until the pointer is null. For example:

```
   IF PCRTVB1L-PTR NOT = NULLS
      SET ADDRESS OF PCRTVB1 TO PCRTVB1L-PTR
      MOVE 'N' TO WS-VB1DONE
   END-IF.

   PERFORM S510-PROCESSVB1 UNTIL VB1DONE.

   S510-PROCESSVB1.
       DISPLAY 'PACKAGE DESCRIPTION : ' PCRTPDSC.
       IF PTR-NEXT-PCRTVB1 NOT = NULLS
            SET ADDRESS OF PCRTVB1 TO PTR-NEXT-PCRTVB1
       ELSE
          MOVE "Y" TO WS-VB1DONE
       END-IF.
```

If the user exit wants to add groups to the list, it is responsible for allocating new storage (using LE facilities, for example, CEECRHP and CEEGTST) and adding to the chain. See member HXCSCHD in the CMNZMF SAMPLES distribution library for an example of how to do this.

If PCRTLOKD is set to YES, data fields on the related panel will be set to output only. The list of panels for which this applies is:

Panel ID	Description	Exit Name
CMNCRT01	Initial package-create panel (non-ERO)	PCRE0001
CMNCRT0R	Initial package-create panel (ERO)	PCRE0001
CMNCRT02	Package description (long form create only)	PCRE0002
CMNCRT03	Installation instructions (long form create only)	PCRE0003
CMNCRT04	Scheduling dependencies	PCRE0004
CMNCRT05	Affected applications	PCRE0005
CMNDPUP1 CMNDPUP2	Package user options	PCRE00PU
CMNCRT06	Install time/date, etc. (ALL site)	PCRE0006
CMNCRT07	Site list with install date/time, etc. (DP site)	PCRE0007
CMNCRT08	Complex/Super information	PCRE0008

PCRTSHRT and PCRTLONG are used to set a message on the next panel/window to be displayed.

If PCRTGO is set to NO, the fields PCRTSHRT, PCRTLONG, and PCRTCURS will be used to set an error message, and the client will (re)display the associated panel.

If you wish to change any of the data fields, do that in place and set PCRTCHNG to YES. If PCRTCHNG is not set to YES, the client ignores any data changes. Note that while the full exit data structure is passed to the exits driven around the package user variable panels (CMNDPUP1/2), only package user variables may be updated by these exits.

From the REXX point of view, the variables making up the data structure have the same names as the equivalent tags in the package-create service. The following extra variables will also be created:

```
dataLocked
proceed
shortMsg
longMsg
cursorField
dataChanged
```

The discussion above identifies the purpose of these variables.

The variable numbers of blocks will be handled by means of indexed stem variables. For example:

```
siteInfo.siteName.
siteInfo.installDate.
siteInfo.fromInstallTime.
siteInfo.toInstallTime.
siteInfo.contactName.
siteInfo.contactPhone.
siteInfo.alternateContactName.
siteInfo.alternateContactPhone.
```

The 0 version of the variable (each variable) will contain the number of instances defined (these will all be the same value). Each level *n* field relates to the same site group for equal *n*.

An example of how to reference them successfully is:

```
Do i = 1 to siteInfo.siteName.0
  say "siteInfo.siteName."||i||" : "siteInfo.siteName.i
end
```

For REXX variable updates the exit can simply set the contents of the variable and then set the dataChanged variable to a value of YES. This indicates to the mainstream client code that it needs to copy the contents of the variables passed back from the exit into whatever local storage it is using for those variables.

A single data structure is passed to all of these exits. The data interface for the package-create exits looks like this:

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
PCRTFUNC	function	8	Function code	No	
PCRTDBUG	debugCall	1	Debug exit call (Y/N)	No	
PCRTORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL	No	
PCRTZMFS	zmfSubs	1	ZMF subsystem Id	No	
PCRTPDB2	db2Subs	4	Primary Db2 subsystem	No	
PCRTUSER	userid	8	User Id	No	
PCRTEXTN	externalName	256	External name for exit	No	
PCRTAPPL	applName	4	Application	No	
PCRTMETH	createMethod	1	Create method	Yes	
PCRTLOKD	dataLocked	3	Fields locked? (YES/NO)	Yes	
PCRTGO	proceed	3	Proceed? (YES/NO)	Yes	
PCRTSHRT	shortMsg	24	Short message	Yes	
PCRTLONG	longMsg	128	Long message	Yes	
PCRTCURS	cursorField	3	Cursor tag	Yes	
PCRTCHNG	dataChanged	3	Data changed? (YES/NO)	Yes	
PCRTCRPK	packageName	10	Created package (post service)	No	
PCRTCFPK	packageModel	10	Name of model package	Yes	001
PCRTPLVL	packageLevel	1	Package level	Yes	002

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
PCRTPTYP	packageType	1	Package type	Yes	003
PCRTRSCD	reasonCode	3	Reason code	Yes	004
PCRTCSPK	complexSuperPackage	10	Complex package	Yes	005
PCRTDEPT	packageDepartment	4	Package department	Yes	006
PCRTNAME	requestorName	25	Requester name	Yes	007
PCRTPHON	requestorPhone	15	Requester phone	Yes	008
PCRTPCAC	problemActionCode	1	Contingency action code	Yes	009
PCRTOPCA	otherProblemAction	44	Other contingency action	Yes	010
PCRTSCHD	schedulerType	1	Scheduler	Yes	011
PCRTTCDU	tempChangeDuration	3	Temporary change duration	Yes	012
PCRTWRQN	packageWorkRequest	12	Work request number	Yes	013
PCRTTITL	packageTitle	255	Package title	Yes	014
PCRTUPAN	userPanel	8	Package user variable panel	Yes	
PCRTOPRF	optsProfile	8	Used to select the ZDDOPTS profile for the display of user options for the ZMF Client Pack	Yes	
Package User Variables					
PCRT01	userVarLen101 - userVarLen115	1 * 15	Set of fifteen 1-byte package user variables	Yes	015-029
PCRT0199	userVarLen199	1	(Reserved)	Yes	
PCRT02	userVarLen201 - userVarLen211	2 * 11	Set of eleven 2-byte package user variables	Yes	030-040
PCRT03	userVarLen301-310	3 * 10	Set of ten 3-byte package user variables	Yes	041-050
PCRT04	userVarLen401-410	4 * 10	Set of ten 4-byte package user variables	Yes	051-060
PCRT08	userVarLen801-810	8 * 10	Set of ten 8-byte package user variables	Yes	061-070
PCRT16	userVarLen1601-1605	16 * 5	Set of five 16-byte package user variables	Yes	071-075
PCRT44	userVarLen4401-4405	44 * 5	Set of five 44-byte package user variables	Yes	076-080
PCRT72	userVarLen7201-7205	72 * 5	Set of five 72-byte package user variables	Yes	081-085
Release and Release Area					

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
PCRTRLSM	release	8	Release	Yes	086
PCRTRARE	releaseArea	8	Release area	Yes	087
Package Description					
PCRTPDSC	packageDesc.*n*	72 * 46	Up to forty-six 72-byte lines of description	Yes	088
Implementation Instructions					
PCRTPIMI	packageImplInst.*n*	72 * 46	Up to forty-six 72-byte lines of implementation instructions	Yes	089
Scheduling Information					
PCRTSSSJ	schedulingInfo. successorJobName.*n*	8	Successor job name	Yes	090
PCRTSSPJ	schedulingInfo. predecessorJobName.*n*	8	Predecessor job name	Yes	091
Participating Packages					
PCRTPPAP PCRTPPNM	partPackageName.*n* (10)	4 6	Set of *n* 4 and 6-byte participating package appl/numbers, *n* 10 byte participating package names (REXX)	Yes	092
Affected Applications					
PCRTAAPP	affectedApplName.*n*	4	Set of *n* 4-byte application names (*n* is limited only by storage constraints)	Yes	093
Install Site Information (Set of *n* sets of installation site information *(n* is limited only by storage constraints)					
PCRTSITE	siteInfo.siteName.*n*	8	Site name	Yes	094
PCRTINDT	siteInfo.installDate.*n*	8	Install date	Yes	095
PCRTFINT	siteInfo. fromInstallTime.*n*	6	Install from time	Yes	096
PCRTTINT	siteInfo.toInstallTime.*n*	6	Install to time	Yes	097
PCRTOANM	siteInfo.contactName.*n*	25	Originating analyst	Yes	098
PCRTOAPH	siteInfo.contactPhone.*n*	15	Analyst phone number	Yes	099
PCRTAANM	siteInfo. alternateContactName. n	25	Alternative analyst	Yes	100
PCRTAAPH	siteInfo. alternateContactPhone. n	15	Alternative analyst phone number	Yes	101

Package Update

This section describes the package-update functional area of the high-level language exits. The 4-character exit name identifier is PUPD.

Select option 3 Package Update from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define customized ISPF variables for the package-update function:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed.

The panels around which exit points will be placed are listed below. The internal exit name (also known as function code) is PUPD0*pnn*, where:

- *p*=0 is the pre-exit.
- *p*=1 is the post-exit.
- *nn* is an alphanumeric identifier relating to the panel for which the exit is taken.

The pre-exit is taken before the panel is displayed (usually to provide model parameters displayed on the panel) and the post-exit is taken after the panel has been displayed (for example, for input validation or enforcement).

Package-update exits are only taken when panels are displayed in input mode.

For package update the package already exists and each update function is updating a specific set of package information. The data format consists of a fixed core of fields, which is provided on every exit call, followed by a single variable length set of information pertaining to the specific update being applied. The map to be used will depend on the exit function code (internal exit name).

Package Update:

Panel ID	Description	Associated Package-Create Panel ID	Exit Name
CMNPGNL1	Updates control information	CMNCRT01	PUPD0p01
CMNPGNL2	Updates package description	CMNCRT02	PUPD0p02

Panel ID	Description	Associated Package-Create Panel ID	Exit Name
CMNPGNL3	Updates installation instructions	CMNCRT03	PUPD0p03
CMNPGNL4	Updates scheduling dependencies	CMNCRT04	PUPD0p04
CMNPGNL5	Updates affected applications	CMNCRT05	PUPD0p05
CMNONSTE	Updates install information	CMNCRT06	PUPD0p06
CMNPRSTI	Updates site install information	CMNCRT07	PUPD0p07
CMNPGNL6	Updates complex/super package information	CMNCRT08	PUPD0p08
CMNDPUP1 CMNDPUP2	Updates package user options		PUPD0pPU
CMNRMBRO	Updates ERO information		PUPD0pER
CMNDB2UP	Updates Db2 option special libtypes		PUPD0pD2
CMNIMSYS	Updates IMS option physical system information		PUPD0pIS
CMNIMACB	Updates IMS options ACB definitions		PUPD0pIA
CMNIMDBD	Updates IMS option DBD definitions		PUPD0pID
CMNIMPSB	Updates IMS option PSB definitions		PUPD0pIP

In order to allow the cursor to be positioned at specific panel fields, a number relating to the panel field in question will be passed back in the XPUPOCURS field. These numbers are documented in the supplied copybook for the function (and, eventually, in the lists of REXX field names). You must make sure that your exit returns a field number that is valid for the panel about to be (re)displayed. If the field number is not valid, a dialog error results.

Each package-update exit call, if it requires variable blocked information, only contains a single block as relevant to the call.

NOTE This is different from package create, which contains them all if they are available, null if not).

```
***
* VARIABLE BLOCK POINTERS
* EACH POINTS TO A DIFFERENT VARIABLE LENGTH SECTION OF DATA.
* THEY APPLY TO EACH SPECIFIC TYPE OF PACKAGE UPDATE.
* THEY REDEFINE THE SAME AREA OF STORAGE AND ARE MUTUALLY
* EXCLUSIVE. PLEASE REFER TO THE VALUE IN PUPDFUNC FOR WHICH
* WHICH MAP TO USE.
*
* EACH ENTRY IN A VARIABLE LENGTH BLOCK CONSISTS OF THE DATA
* FOLLOWED BY A POINTER TO THE NEXT ENTRY. WHEN THAT POINTER
* IS NULL THEN THERE ARE NO FURTHER ENTRIES IN THE BLOCK.
***
     03 PUPDVARB-PTR        USAGE IS POINTER.

***
* PUPDFUNC = 'PUPD0002' OR 'PUPD0102'
*
* PACKAGE DESCRIPTION - UP TO 46 LINES OF 72 BYTES
***
 01  PUPDVB1.
     03 PUPDPDSC            PIC X(72).
*                                     LINE OF DESCRIPTION (087)
     03  PTR-NEXT-PUPDVB1   POINTER.
*                                     POINTER TO NEXT LINE
***
* PUPDFUNC = 'PUPD0003' OR 'PUPD0103'
*
* PKG IMPLEMENTATION INSTRUCTIONS - UP TO 46 LINES OF 72 BYTES
***
 01  PUPDVB2 REDEFINES PUPDVB1.
     03 PUPDPIMI            PIC X(72).
*                                     PKG IMPL INSTRUCTION(088)
     03  PTR-NEXT-PUPDVB2   POINTER.
*                                     POINTER TO NEXT LINE
.
.
.
```

The user exit should follow the pointer chains for the single repeating group until the pointer is null.

If PUPDLOKD is set to YES, data fields on the related panel will be set to output only. The list of panels for which this applies is:

Panel ID	Description	Associated Package-Create Panel ID	Exit Name
CMNPGNL1	Updates control information	CMNCRT01	PUPD0001
CMNPGNL2	Updates package description	CMNCRT02	PUPD0002
CMNPGNL3	Updates installation instructions	CMNCRT03	PUPD0003
CMNPGNL4	Updates scheduling dependencies	CMNCRT04	PUPD0004
CMNPGNL5	Updates affected applications	CMNCRT05	PUPD0005
CMNONSTE	Updates install information	CMNCRT06	PUPD0006
CMNPRSTI	Updates site install information	CMNCRT07	PUPD0007

Panel ID	Description	Associated Package-Create Panel ID	Exit Name
CMNPGNL6	Updates complex/super package information	CMNCRT08	PUPD0008
CMNDB2UP	Updates Db2 option special libtypes		PUPD00D2
CMNIMSYS	Updates IMS option physical system information		PUPD00IS
CMNIMACB	Updates IMS options ACB definitions		PUPD00IA
CMNIMDBD	Updates IMS option DBD definitions		PUPD00ID
CMNIMPSB	Updates IMS option PSB definitions		PUPD00IP

PUPDSHRT and PUPDLONG are used to set a message on the next panel/window to be displayed.

If PUPDGO is set to NO, the fields PUPDSHRT, PUPDLONG, and PUPDCURS will be used to set an error message and the associated panel will be (re)displayed by the client.

If the user exit wishes to change any of the data fields, it does that in place and sets PUPDCHNG to YES. If PUPDCHNG is not set to YES, any data changes are ignored by the client. Note that while the full exit data structure is passed to the exits driven around the package user variable panels (CMNDPUP1/2) only package user variables may be updated by these exits.

A single data structure is passed to all of these exits. The data interface for the package-update exits looks like this:

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
PUPDFUNC	function	8	Function code	No	
PUPDDBUG	debugCall	1	Debug exit call (Y/N)	No	
PUPDORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL	No	
PUPDZMFS	zmfSubs	1	ZMF subsystem Id	No	
PUPDPDB2	db2Subs	4	Primary Db2 subsystem	No	
PUPDUSER	userid	8	User Id	No	
PUPDEXTN	externalName	256	External name for exit	No	
PUPDAPPL	applName	4	Application	No	
PUPDLOKD	dataLocked	3	Fields locked? (YES/NO)	Yes	

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
PUPDGO	proceed	3	Proceed? (YES/NO)	Yes	
PUPDSHRT	shortMsg	24	Short message	Yes	
PUPDLONG	longMsg	128	Long message	Yes	
PUPDCURS	dataChanged	3	Cursor tag	Yes	
PUPDCHNG	cursorField	3	Data changed? (YES/NO)	Yes	
PUPDPKGN	packageName	10	Package name	No	
PUPDCTSI	packageCreator	8	User id of package creator	No	
PUPDPLVL	packageLevel	1	Package level	Yes	001
PUPDPTYP	packageType	1	Package type	Yes	002
PUPDPSTT	packageStatus	1	Package status	No	
PUPDCSPK	complexSuperPackage	10	Complex package	Yes	003
PUPDCSPS	complexSuperPkgStatus	1	Complex package status	No	
PUPDRSCD	reasonCode	3	Reason code	Yes	004
PUPDDEPT	packageDepartment	4	Department	Yes	005
PUPDNAME	requestorName	25	Requester name	Yes	006
PUPDPHON	requestorPhone	15	Requester phone	Yes	007
PUPDPCAC	problemActionCode	1	Contingency action code	Yes	008
PUPDOPCA	otherProblemAction	44	Other contingency action	Yes	009
PUPDSCHD	schedulerType	1	Scheduler	Yes	010
PUPDTCDU	tempChangeDuration	3	Temporary change duration	Yes	011
PUPDWRQN	packageWorkRequest	12	Work request number	Yes	012
PUPDTITL	packageTitle	255	Package title	Yes	013
PUPDUPAN	userPanel	8	Package user variable panel	Yes	
PUPDOPRF	optsProfile	8	Used to select the ZDDOPTS profile for the display of user options for the ZMF Client Pack	Yes	
Package User Variables					
PUPD01	userVarLen101 - userVarLen115	1 * 15	Set of fifteen 1-byte package user variables	Yes	014-028
PUPD0199	userVarLen199	1	(Reserved)	Yes	

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
PUPD02	userVarLen201 - userVarLen211	2 * 11	Set of eleven 2-byte package user variables	Yes	029-039
PUPD03	userVarLen301-310	3 * 10	Set of ten 3-byte package user variables	Yes	040-049
PUPD04	userVarLen401-410	4 * 10	Set of ten 4-byte package user variables	Yes	050-059
PUPD08	userVarLen801-810	8 * 10	Set of ten 8-byte package user variables	Yes	060-069
PUPD16	userVarLen1601-1605	16 * 5	Set of five 16-byte package user variables	Yes	070-074
PUPD44	userVarLen4401-4405	44 * 5	Set of five 44-byte package user variables	Yes	075-079
PUPD72	userVarLen7201-7205	72 * 5	Set of five 72-byte package user variables	Yes	080-084
Release and Release Area					
PUPDRLSM	release	8	Release	Yes	085
PUPDRARE	releaseArea	8	Release area	Yes	086
Package Description					
PUPDPDSC	packageDesc.*n*	72 * 46	Up to forty-six 72-byte lines of description	Yes	087
Implementation Instructions					
PUPDPIMI	packageImplInst.*n*	72 * 46	Up to forty-six 72-byte lines of implementation instructions	Yes	088
Scheduling Information					
PUPDSSSJ	schedulingInfo. successorJobName.*n*	8	Successor job name	Yes	089
PUPDSSPJ	schedulingInfo. predecessorJobName.*n*	8	Predecessor job name	Yes	090
Participating Packages					
PUPDPPAP PUPDPPNM	partPackageName.*n* (10)	4 6	Set of *n* 4 and 6-byte participating package appl/numbers (LE), n 10 byte participating package names (REXX)	Yes	091
Affected Applications					
PUPDAAPP	affectedApplName.n	4	Set of *n* 4-byte application names (*n* is limited only by storage constraints)	Yes	092

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
Install Site Information (Set of *n* sets of installation site information *(n* is limited only by storage constraints)					
PUPDSITE	siteInfo.siteName.*n*	8	Site name	Yes	093
PUPDINDT	siteInfo.installDate.*n*	8	Install date	Yes	094
PUPDFINT	siteInfo. fromInstallTime.*n*	6	Install from time	Yes	095
PUPDTINT	siteInfo.toInstallTime.*n*	6	Install to time	Yes	096
PUPDOANM	siteInfo.contactName.*n*	25	Originating analyst	Yes	097
PUPDOAPH	siteInfo.contactPhone.*n*	15	Analyst phone number	Yes	098
PUPDAANM	siteInfo. alternateContactName.*n*	25	Alternating analyst	Yes	099
PUPDAAPH.n	siteInfo. alternateContactPhone.*n*	15	Analyst phone number	Yes	100
Db2 Libtype Information (Set of *n* sets of Db2 libtype information (*n* is limited only by storage constraints)					
PUPDDLTP	db2Info.libType.*n*	3	Db2 Libtype	Yes	101
PUPDDSUB	db2Info.subType.*n*	1	Db2 Sub type	Yes	102
PUPDEOSC	db2SqlTerminationChar. *n*	1	SQL end of sentence	Yes	103
IMS System Information (Set of *n* sets of IMS system information (*n* is limited only by storage constraints)					
PUPDISID	imsSysInfo.imsControl Region.*n*	4	IMS system Id	Yes	104
PUPDISRS	imsSysInfo.imsSiteNam e.*n*	8	ZMF remote site	Yes	105
PUPDISLN	imsSysInfo.imsLogicalS ite.*n*	8	IMS logical site name	Yes	106
PUPDISOP	imsSysInfo.isImsGlobal ActivationEnabled.*n*	1	IMS option activated	Yes	107
PUPDISDV	imsSysInfo.imsDevCha rSuffix.*n*	1	IMS device character	Yes	108
PUPDISMF	imsSysInfo.isMfsAlways Generated.*n*	1	Process MFS (Y/N)?	Yes	109
PUPDISPS	imsSysInfo.isPsbAlways Generated.*n*	1	Process PSB (Y/N)?	Yes	110
PUPDISDB	imsSysInfo.isDbdAlway sGenerated.*n*	1	Process DBD (Y/N)?	Yes	111
PUPDISAC	imsSysInfo.isAcbAlway sCreatedForPcbs.*n*	1	Process ACB (Y/N)?	Yes	112
PUPDISBU	imsSysInfo.imsBackup ModelLib.*n*	25	IMS back-up model data set name	Yes	113

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
PUPDISS1	imsSysInfo.imsGenMacroComponent.*n*	8	Stage 1 gen member name	Yes	114
PUPDISRL	imsSysInfo.imsResLib.*n*	44	IMS RESLIB	Yes	115
PUPDISML	imsSysInfo.imsMacLib.*n*	44	IMS MACLIB	Yes	116
PUPDISMS	imsSysInfo.imsModStatLib.*n*	44	IMS MODSTAT	Yes	117
PUPDISGM	imsSysInfo.imsGenMacroStageLib.*n*	44	IMS sysgen MACLIB	Yes	118
PUPDISPL	imsSysInfo.imsPsbLib.*n*	44	IMS PSBLIB	Yes	119
PUPDISDL	imsSysInfo.imsDbdLib.*n*	44	IMS DBDLIB	Yes	120
PUPDISAL	imsSysInfo.imsAcbLib.n	44	IMS ACBLIB	Yes	121
PUPDISFL	imsSysInfo.imsFormatLib.*n*	44	IMS FMTLIB	Yes	122
PUPDISRF	imsSysInfo.imsRefLib.*n*	44	IMS referal	Yes	123
IMS ACB Information (Set of *n* sets of IMS system information (*n* is limited only by storage constraints)					
PUPDIAID	imsAcbInfo.imsControlRegion.*n*	4	IMS system Id	Yes	124
PUPDIARS	imsAcbInfo.imsSiteName.*n*	8	ZMF remote site	Yes	125
PUPDIALN	imsAcbInfo.imsLogicalSite.*n*	8	IMS logical site name	Yes	126
PUPDIATY	imsAcbInfo.acbStatementType.*n*	3	IMS ACB type	Yes	127
PUPDIACT	imsAcbInfo.acbGenStatementType.*n*	8	ACB control word	Yes	128
PUPDIASR	imsAcbInfo.component.*n*	8	ACB source name	Yes	129
PUPDIATG	imsAcbInfo.targetComponentType.*n*	8	ACB target name	Yes	130
PUPDIALT	imsAcbInfo.componentType.*n*	3	ACB libtype	Yes	131
PUPDIATL	imsAcbInfo.targetComponentType.*n*	3	ACB target libtype	Yes	132
IMS PSB/DBD Information (Set of *n* sets of IMS system information (*n* is limited only by storage constraints)					
PUPDIPID	imsDbdPsbInfo.imsControlRegion.*n*	4	IMS system Id	Yes	133
PUPDIPRS	imsDbdPsbInfo.imsSiteName.*n*	8	ZMF remote site	Yes	134
PUPDIPLN	imsDbdPsbInfo.imsLogicalSite.*n*	8	IMS logical site name	Yes	135

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
PUPDIPCT	imsDbdPsbInfo.controlStatement.*n*	8	Control word	Yes	136
PUPDIPSR	imsDbdPsbInfo.component.*n*	8	Source name	Yes	137
PUPDIPLT	imsDbdPsbInfo.componentType.*n*	3	Libtype	Yes	138
PUPDIPOV	imsDbdPsbInfo.overrideStatement.*n*	64	Override statement	Yes	139
PUPDIPOR	imsDbdPsbInfo.originalStatement.*n*	64	Original statement	Yes	140

File Tailoring

The file tailoring function is somewhat different to the usual HLLX process in that there is no user interaction to work with. The file tailoring function provides a mechanism for other HLL exits to set variable values that the file tailoring programs can pick up and turn in to ISPF variables that are used in file tailoring the skeletons.

The 4-character exit name identifier for file tailoring is FTLR.

Select option 5 File Tailoring from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define customized ISPF variables for file tailoring:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed:

```
CMNHLLMN                      HLL Exit Definition             Row 1 to 5 of 5
Command ===> ____________________________________________ Scroll ===> CSR

Internal External  + Active  1=LE    Description  +
  Name     Name              2=REXX  Debug Userids  +
-------------------------------------------------------------------------------
FTLR00BA _________     NO     2       install/baseline file tailoring
 Debug:  _________     NO     2       ____________________________________
-------------------------------------------------------------------------------
FTLR00BL _________     NO     2       build job file tailoring
 Debug:  _________     NO     2       ____________________________________
-------------------------------------------------------------------------------
FTLR00BP _________     NO     2       base ZMF promotion file tailoring
 Debug:  _________     NO     2
-------------------------------------------------------------------------------
FTLR00EB _________     NO     2       ERO autoresolve file tailoring
 Debug:  _________     NO     2
-------------------------------------------------------------------------------
FTLR00EP _________     NO     2       ERO promotion file tailoring
 Debug:  _________     NO     2
```

There is a single exit point per file tailoring program:

File Tailoring Program Name	Description	Exit Name
CMNVFTLR	ZMF component build	FTLR00BL
CMNVPRFT	ZMF promote/demote	FTLR00BP
CMNVPIJB	ZMF install/baseline x.node	FTLR00BA
CMNVRACR	ERO autoresolve build	FTLR00EB
CMNVRPFT	ERO promote/demote	FTLR00EP

The single exit point is called multiple times (hard limited to 999 to avoid loops) at the control of the exit code (that is, the exit code is called until the exit says stop calling me). Each time the exit can pass back a single variable name, length, and value. The file tailoring program vdefines and assigns a value to each variable that the exit passes to it.

A REXX sample, in which variables passed to the exit are displayed and three ISPF variables are defined, is supplied as member HXRFTLR of the CMNZMF.SAMPLES distribution library. A COBOL sample is supplied as member HXCFTLR of the CMNZMF.SAMPLES distribution library.

A single data structure is passed to all of these exits. Many of the fields are related to component build only (as detailed below). There is, potentially, a huge number of different fields that may be required by the promote/demote and baseline/install exits in order to make a decision on what, if any, extra ISPF variables need to be defined. It is expected that the exit can use ZMF XML services to access that information.

We envisage the way to use these exits is to have other exits (for example, exits for package create and/or package update) to set certain VPOOL variables which can then be subsequently accessed by the file tailoring exits and turned into ISPF variables for use in the target file tailoring skeletons.

The data interface for the file tailoring exits looks like this:

LE-Language Variable Name	REXX Variable Name	Length	Purpose
General			
FTLRFUNC	function	8	Internal exit name
FTLRDBUG	debugCall	1	Debug exit call (Y/N)
FTLRORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL
FTLRZMFS	zmfSubs	1	ZMF subsystem character
FTLRPDB2	db2Subs	4	Default Db2 subsystem for this ZMF instance
FTLRUSER	userid	8	Userid for function calling this exit
FTLREXTN	externalName	256	External routine name defined for this exit

LE-Language Variable Name	REXX Variable Name	Length	Purpose
FTLRVCNT	cumulativeVarCount	3	How many times has this exit been called
FTLRLAST	lastVarName	8	The name of the last variable defined by ZMF at the request of this exit
FTLRFROM	fromEroZmf	3	Called from an ERO or base ZMF process
FTLRPKGN	packageId	10	Package name
FTLRJOB1	jobCard01	72	Job card line #1 (blank for Install/Baseline)
FTLRJOB2	jobCard02	72	Job card line #2 (blank for Install/Baseline)
FTLRJOB3	jobCard03	72	Job card line #3 (blank for Install/Baseline)
FTLRJOB4	jobCard04	72	Job card line #4 (blank for Install/Baseline)
FTLRRLSE	release	8	ERO release
FTLRAREA	releaseArea	8	ERO area
FTLRUV1	userVariable01 userVariable02 userVariable03 userVariable04 userVariable05	8*5	Set of five 8-byte user variables
FTLRUV6	userVariable06 userVariable07 userVariable08 userVariable09 userVariable10	72*5	Set of five 72-byte user variables
Promotion/Demotion			
FTLRPSIT	promoSite	8	Target site
FTLRPLVL	promoLevel	2	Target level *nn*
FTLRPNME	promoName	8	Target promotion name
FTLRPFUN	promoFunction	8	'PROMOTE' 'SELPROM' 'CLEANUP' 'DEMOTE' 'SELDEMO'
Build			
FTLRVOBJ	buildProcessObject	8	'cmponent'
FTLRVMSG	buildProcessMessage	8	'submit'

LE-Language Variable Name	REXX Variable Name	Length	Purpose
FTLRVSCP	buildProcessScope	8	'stage' 'recomp' 'relink' 'checkout' 'build' 'checkin'
FTLRVPRC	buildProcessName	8	Internal name for file tailoring process (usually VCOMP000).
FTLRLTYP	componentType	3	Library type
FTLRCOMP	component	256	Component name
FTLRPROC	buildProc	8	Build procedure
FTLRLANG	language	8	Language
FTLRDB2P	useDb2PreCompileOption	1	Db2 precompile requested (Y/N)
FTLRUO1	userOption01 userOption02 userOption03 userOption04 userOption05 userOption06 userOption07 userOption08 userOption09 userOption10	1*10	First set of the original component user options
FTLRUO2	userOption11 userOption12 userOption13 userOption14 userOption15 userOption16 userOption17 userOption18 userOption19 userOption20	1*10	Second set of the original component user options
FTLR01	userOption0101 userOption0102 userOption0103 userOption0104 userOption0105	1*5	Set of five extended 1-byte component user options
FTLR02	userOption0201 userOption0202 userOption0203	2*3	Set of three extended 2-byte component user options
FTLR03	userOption0301 userOption0302 userOption0303	3*3	Set of three extended 3-byte component user options
FTLR04	userOption0401 userOption0402 userOption0403	4*3	Set of three extended 4-byte component user options

LE-Language Variable Name	REXX Variable Name	Length	Purpose
FTLR08	userOption0801 userOption0802 userOption0803 userOption0804 userOption0805	8*5	Set of five extended 8-byte component user options
FTLR10	userOption1001 userOption1002	10*2	Set of two extended 10-byte component user options
FTLR16	userOption1601 userOption1602	16*2	Set of two extended 16-byte component user options
FTLR34	userOption3401 userOption3402	34*2	Set of two extended 34-byte component user options
FTLR44	userOption4401 userOption4402	44*2	Set of two extended 44-byte component user options
FTLR64	userOption6401 userOption6402 userOption6403 userOption6404 userOption6405	64*5	Set of five extended 64-byte component user options
FTLR72	userOption7201 userOption7202 userOption7203 userOption7204 userOption7205	72*5	Set of five extended 72-byte component user options
Returned by exit			
FTLRGO	proceed	3	Set to 'NO' to stop the file tailoring process
FTLRSHRT	shortMsg	24	Not used at present
FTLRLONG	longMsg	128	Set to message text you wish to be reported in the file tailoring task output when stopping the process
FTLRMORE	moreToCome	3	Set to 'YES' if you wish to define another variable. ZMF will keep calling this exit until this is set to something other than 'YES' (max 999 times)
FTLRVNAM	ispfVarName	8	The name of the ISPF variable you wish to have ZMF vdefine for use by the current file tailoring process
FTLRVLEN	ispfVarLen	4	The length of the variable value
FTLRVVAL	ispfVarValue	1024	The variable value (character variables only)

The following example, taken from the HXRFTLR member of the CMNZMF.SAMPLES distribution library, shows how to have this exit define three ISPF variables:

```
/*                                                                    */
/* Here we define three variables which will be available during the  */
/* skeleton file tailoring performed by the current process.          */
/* This exit is able to return a single variable at a time and is     */
/* called repeatedly while moreToCome is set to YES (max 999 times).  */
/*                                                                    */
/* The name of the last variable to be defined by this exit is        */
/* presented on the next call in the lastVarName variable.            */
/*                                                                    */
If lastVarName = "        " then
  Do
    ispfVarName  = "MYVAR1"
    ispfVarLen   = "12"
    ispfVarValue = "1234567890AB"
    moreToCome   = "YES"
  End

If lastVarName = "MYVAR1" then
  Do
    ispfVarName  = "MYVAR2"
    ispfVarLen   = "8"
    ispfVarValue = "12345678"
    moreToCome   = "YES"
  End

If lastVarName = "MYVAR2" then
  Do
    ispfVarName  = "MYVAR3"
    ispfVarLen   = "4"
    ispfVarValue = "1234"
    moreToCome   = "NO"
  End
```

The following example shows how to stop the process and provide a reason for doing so:

```
/*                                                                    */
/* stopping the file tailoring process and setting an error message   */
/*                                                                    */
proceed = "NO"
longMsg = "Example of how to let the developer know what went wrong"
```

Checkout

This section describes the checkout functional area of the high-level language exits. The 4-character exit name identified is CKOT.

Select option 6 Checkout from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define exits for component checkout from baseline/promotion:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed. Here is how a sample panel might look:

```
CMNHLLMN                      HLL Exit Definition           Row 1 to 12 of 13
Command ===> ____________________________________________________ Scroll ===> CSR

 Internal External  + Active  1=LE    Description  +
  Name     Name              2=REXX  Debug Userids  +
-------------------------------------------------------------------------------
CKOT01CK _________      NO     2       post checkout entry panel
 Debug:  CKOTALL        YES    2       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT01CL _________      NO     2       post component list for pkg ckot
 Debug:  CKOTALL        YES    2       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT01DL _________      NO     2       post component delete
 Debug:  CKOTALL        YES    2       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT01LB _________      NO     2       post library list where comp found
 Debug:  CKOTALL        YES    2       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT01LT _________      NO     2       post libtype display
 Debug:  CKOTALL        YES    2       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT01MS _________      NO     2       post member selection list
 Debug:  CKOTALL        YES    2       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT01PL _________      NO     2       post promotion library list
 Debug:  HXPCKOT        YES    1       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT00XM _________      NO     2       pre  service call
 Debug:  CKOTALL        NO     2       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT01XM _________      NO     2       post service call
 Debug:  CKOTALL        YES    2       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT0001 _________      NO     2       pre  checkout selection criteria
 Debug:  HXCCKOT        NO     1       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT0101 _________      NO     2       post checkout selection criteria
 Debug:  HXCCKOT        YES    1       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT0002 _________      NO     2       pre  batch checkout panel
 Debug:  CKOJOBCD       YES    2       USERA1  ,USERA2
-------------------------------------------------------------------------------
CKOT0102 _________      NO     2       post batch checkout panel
 Debug:  CKOJOBCD       YES    2       USERA1  ,USERA2
```

The panels around which exit points will be placed are listed below. The internal exit name (also known as function code) is CKOT0*pnn*, where:

- *p*=0 is the pre-exit.
- *p*=1 is the post-exit.
- *nn* is an alphanumeric identifier relating to the panel for which the exit is taken.

An internal exit name of CKOT0*p*01, for example, means that both pre- and post-exits exist. That is, the name of the pre-exit is CKOT0001 and the name of the post-exit is CKOT0101. If it makes no sense to have a pre-exit, the internal exit name is given as CKOT0101 (post-exit only). If it makes no sense to have a post-exit, the internal exit name is given as CKOT0001 (pre-exit only).

The pre-exit is taken before the panel is displayed and the post-exit is taken after the panel has been displayed.Most table displays have only post-exits. That is, we do not want to have a pre-exit that manipulates the lists that ZMF generates. We may want to have a post-exit to validate the selections that the user makes from the lists.

The panels around which the checkout exit points are placed are:

Panel Id	Description	Exit Name
CMNMCKOT/R	Checkout entry panel from the build function	CKOT01CK
CMNCKOT1	Checkout selection criteria	CKOT0001 CKOT0101
CMNCKOT2	Batch checkout panel	CKOT0002 CKOT0102
CMNCLTSL	Libtype table display	CKOT01LT
CMNCCMSL/2	Member list baseline/promotion	CKOT01MS
CMNCMLSL	Member locate library list	CKOT01LB
CMNCPLSL	Promotion library list	CKOT01PL
CMNCKOTS/L/X	Member list - package checkout	CKOT01CL
CMNSTG20	Confirm delete request	CKOT01DL

Pre- and post-XML-service calls for checkout are:

XML Service Name	Description	Exit Name
cmponent.service.checkout	Component Checkout	CKOT00XM CKOT01XM

Sample exits are provided in the CMNZMF.SAMPLES distribution library. These examples show how to list all the information coming in to the exits. Not all information is available to all exits. The exits that occur early in the dialog will not have as much information as the exits that occur later in the dialog.

The checkout exit examples are:

CMNZMF.SAMPLES Library Member	Description
HXCCKOT	COBOL example
HXPCKOT	PL/I example
HXRCKOT	REXX example

A single data structure is passed to all of these exits. The data interface looks like this:

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
CKOTFUNC	function	8	Internal exit name	No	
CKOTDBUG	debugCall	1	Debug exit call (Y/N)	No	

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
CKOTORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL	No	
CKOTZMFS	zmfSubs	1	ZMF subsystem character	No	
CKOTPDB2	db2Subs	4	Default Db2 subsystem for this ZMF	No	
CKOTUSER	userid	8	Userid for function calling this exit	No	
CKOTEXTN	externalName	256	External routine name defined for this exit	No	
CKOTPKGN	packageId	10	The package being acted on	No	
CKOTPSTA	packageStatus	3	Package status (DEV, FRZ, and so on)	No	
CKOTPINS	packageInsDate	8	Package Install Date yyyymmdd	No	
CKOTCOMP	componentName	256	Component Name	Yes	001
CKOTLTYP	componentType	3	Component library type	Yes	002
CKOTBSLB	chkOutSourceLocation	1	Source Location	Yes	
CKOTMODE	chkOutMode	1	Online or Batch	Yes	008
CKOTBLVL	basePromoLibLevel	3	Baseline/Promotion level	Yes	003
CKOTPSIT	promotionSiteName	8	Promotion Site	Yes	013
CKOTPNAM	promotionName	8	Promotion Name	Yes	014
CKOTRLSN	release	8	Release Name	Yes	
CKOTAREA	releaseArea	8	Release Area	Yes	
CKOTCKTO	chkOutTargetLocation	1	Target Location (dev/stage)	Yes	004
CKOTCKDS	personalLibStorageMeans	1	Personal lib dsorg	Yes	006
CKOTPDSN	personalLib	1026	Personal library name	Yes	005
CKOTLOCK	lockComponent	3	Lock Component (Yes/No)	Yes	009
CKOTUCUO	useCompUsrOpts	3	Use component user options (Yes/No)	Yes	018
CKOTSPSV	savePriorVersion	3	Save staging versions (Yes/No)	Yes	
CKOTSUPN	suppressNotify	3	Suppress batch messages (Yes/No)	Yes	024
CKOTMIXC	mixedCase	1	Name has mixed case?	Yes	037
CKOTJOB1	jobCard01	72	Job card line #1	Yes	020
CKOTJOB2	jobCard02	72	Job card line #2	Yes	021

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
CKOTJOB3	jobCard03	72	Job card line #3	Yes	022
CKOTJOB4	jobCard04	72	Job card line #4	Yes	023
CKOTUVPN	userVarPanel	8	User variable panel name	Yes	
CKOTUV01 - 05	userVariable01 userVariable02 userVariable03 userVariable04 userVariable05	8*5	Set of five 8-byte package user variables	Yes	027 028 029 030 031
CKOTUV06 - 10	userVariable06 userVariable07 userVariable08 userVariable09 userVariable10	72*5	Set of five 72-byte package user variables	Yes	032 033 034 035 036
CKOTSLTP	selLibraryType	3	Selected library type	No	
CKOTVVMM	verModLevel	5	Version.mod level	No	
CKOTCRDT	createDate	10	Member create date	No	
CKOTCHGD	changeDate	10	Member change date	No	
CKOTCHGT	changeTime	5	Member change time	No	
CKOTCSZE	memberSize	5	Member change size	No	
CKOTUSRN	username	8	User name	No	
CKOTLSZE	loadSize	6	Load member size	No	
CKOTLTTR	loadTtr	6	Load member TTR	No	
CKOTALAS	loadAlias	8	Load member alias	No	
CKOTSSSI	loadSetssi	8	Load member setssi	No	
CKOTATTR	loadAttr	8	Load member attributes	No	
CKOTGO	proceed	3	Set to 'NO' to stop the process	Yes	
CKOTLOKD	dataLocked	3	Fields locked? (YES/NO)	Yes	
CKOTSHRT	shortMsg	24	Short error message text	Yes	
CKOTLONG	longMsg	128	Long error message text	Yes	
CKOTOPTN	optionRequested	1	Option used in ISPF, either from the primary entry panel for checkout or the digit in the C1, C2, C3, C4 checkout line commands available from the package list	No	
CKOTCURS	cursorField	3	For ISPF where you wish the cursor to be placed on return to the panel display. The values relating to each field are shown in this table.	Yes	

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
CKOTCHNG	dataChanged	3	This field must be set to YES if you wish to return changed values to ZMF.	Yes	
CKOTCOVL	confirmOverlay	3	This field must be set to YES if you wish confirm overlay.	Yes	019
CKOTUVAR	userVariables	1	Display User variable panel (Y/N)	Yes	026
CKOTSPKG	sourcePackage	10	Checkout from this package	Yes	025
CKOTOPRF	optsProfile	8	Used to select the ZDDOPTS profile for the display of user options for the ZMF Client Pack	Yes	

Promote/Demote

This section describes the promote/demote functional area of the high-level language exits. The 4-character exit name identifier is PRDM.

Select option 7 Promote/Demote from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define exits for component promotion and demotion:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed. Here is how a sample panel might look:

```
CMNHLLMN                      HLL Exit Definition            Row 1 to 6 of 12
Command ===> ____________________________________________ Scroll ===> CSR

Internal External  + Active  1=LE    Description  +
  Name     Name              2=REXX  Debug Userids  +
-------------------------------------------------------------------------------
PRDM00XD _________     NO     2       pre  demotion  service _________________
 Debug:  _________     NO     2       ________________________________________
-------------------------------------------------------------------------------
PRDM01XD _________     NO     2       post demotion  service _________________
 Debug:                NO     2       ________________________________________
-------------------------------------------------------------------------------
PRDM00XP _________     NO     2       pre  promotion service _________________
 Debug:  _________     NO     2       ________________________________________
-------------------------------------------------------------------------------
PRDM01XP _________     NO     2       post promotion service _________________
 Debug:  _________     NO     2       ________________________________________
-------------------------------------------------------------------------------
PRDM0100 _________     NO     2       post promote/demote main menu __________
 Debug:  _________     NO     2       ________________________________________
-------------------------------------------------------------------------------
PRDM0101 _________     NO     2       post site selection ____________________
 Debug:  _________     NO     2       ________________________________________
    ...
```

The panels around which exit points will be placed are listed below. The internal exit name (also known as function code) is PRDM0*pnn*, where:

- *p*=0 is the pre-exit.

- *p*=1 is the post-exit.
- *nn* is an alphanumeric identifier relating to the panel for which the exit is taken.

The pre-exit is taken before the panel is displayed and the post-exit is taken after the panel has been displayed.

An internal exit name of PRDM0*p*01, for example, means that both pre- and post-exits exist. That is, the name of the pre-exit is PRDM0001 and the name of the post-exit is PRDM0101. If it makes no sense to have a pre-exit, the internal exit name is given as PRDM0101 (post-exit only). If it makes no sense to have a post-exit, the internal exit name is given as PRDM0001 (pre-exit only).

Most table displays have only post-exits. That is, we do not want to have a pre-exit that manipulates the lists that ZMF generates. We may want to have a post-exit to validate the selections that the user makes from the lists.

The panels around which the promote/demote exit points are placed are:

Panel Id	Description	Exit Name
CMNRPM00	Promote/demote main menu	PRDM0100
CMNRPM01	Site selection	PRDM0101
CMNRPM03	Promote options	PRDM0003 PRDM0103
CMNRPM04	Demote options	PRDM0004 PRDM0104
CMNRPM05	Selective promote/demote	PRDM0105
CMNRPM07	Promotion level selection	PRDM0107

Pre- and post-XML-service calls for promote and demote are:

XML Service Name	Description	Exit Name
package.service.promote	Component promote	PRDM00XP PRDM01XP
package.service.demote	Component demote	PRDM00XD PRDM01XD

Sample exits are provided in the CMNZMF.SAMPLES distribution library. These examples show how to list all the information coming in to the exits. Not all information is available to all exits. The exits that occur early in the dialog will not have as much information as the exits that occur later in the dialog.

The promote/demote exit examples are:

CMNZMF.SAMPLES Library Member	Description
HXCPRDM	COBOL example
HXPPRDM	PL/I example
HXRPRDM	REXX example

Most of the fields are fixed in nature. (See the data interface below.) However, there is an optional variable length section. This section contains the names and library types of all the components selected for promotion/demotion.

The format of this variable length data in COBOL is:

```
***
* VARIABLE BLOCK POINTER
*
* EACH ENTRY IN A VARIABLE LENGTH BLOCK CONSISTS OF THE DATA
* FOLLOWED BY A POINTER TO THE NEXT ENTRY. WHEN THAT POINTER
* IS NULL THEN THERE ARE NO FURTHER ENTRIES IN THE BLOCK.
***
     03 PRDMVARB-PTR        USAGE IS POINTER.
*
***
* SELECTED COMPONENT LIST
***
 01  PRDMCPNT.
     03  PTR-NEXT-PRDMCPNT  POINTER.
*                                    POINTER TO NEXT ENTRY
     03 PRDMCTYP            PIC X(3).
*                                    THE CMPNT LIBTYPE
     03 PRDMCOMP.
*                                    THE VARLEN COMPONENT NAME
        49 PRDMCOMP-LEN     PIC S9(4)  COMP.
*                                                  LENGTH
        49 PRDMCOMP-NAME    PIC X OCCURS 0 TO 256 TIMES
                                  DEPENDING ON PRDMCOMP-LEN.
*                                                  NAME
```

The format of this variable length data in PL/I is:

```
/***                                                          */
/* VARIABLE BLOCK POINTERS                                    */
/*                                                            */
/* EACH ENTRY IN A VARIABLE LENGTH BLOCK CONSISTS OF THE DATA  */
/* FOLLOWED BY A POINTER TO THE NEXT ENTRY. WHEN THAT POINTER  */
/* IS NULL THEN THERE ARE NO FURTHER ENTRIES IN THE BLOCK.     */
/***                                                          */
       2 PRDMVBP               PTR;
/***                                                          */
/* SELECTED COMPONENT LIST                                    */
/***                                                          */
DCL     1 PRDMCPNT             BASED(WORKVBP),
            2 PTR_NEXT_PRDMCPNT PTR,      /*POINTER TO NXT BLOCK*/
            2 PRDMCTYP          CHAR(3),  /*COMPONENT LIBTYPE   */
            2 PRDMCOMP          CHAR(256) VARYING; /* CMPNT NAME*/
```

The method for traversing this variable length list is the same as that used in other functions. The anchor pointer (PRDMVBP) points to the first in the chain of entries (null if no chain exists). Each entry contains a pointer to the next entry (null at end of chain).

The difference with this list to other functions is that the data in each entry is variable in length itself. (The component name can be any length up to 256 bytes.) The sample code shows methods for dealing with this variable length.

REXX makes use of stem variables as usual with a variable number of similar data items.

Note that the list of components is only provided on a selective promote/demote and consists of just those components selected. On a full promote/demote no such list is provided. (The list is potentially huge and may never be needed.) If such information is needed, you can easily obtain it in the exit itself by making use of ZMF XML services. The samples provided show how this is done.

A single data structure is passed to all of these exits. The data interface looks like this:

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
PRDMFUNC	function	8	Internal exit name	No	
PRDMDBUG	debugCall	1	Debug exit call (Y/N)	No	
PRDMORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL	No	
PRDMZMFS	zmfSubs	1	ZMF subsystem character	No	
PRDMPDB2	db2Subs	4	Default Db2 subsystem for this ZMF instance	No	
PRDMUSER	userid	8	Userid for function calling this exit	No	
PRDMEXTN	externalName	256	External routine name defined for this exit	No	
PRDMPKGN	packageId	10	The package being acted on	No	001
PRDMPSTA	packageStatus	3	Package status (DEV, FRZ, and so on)	No	002
PRDMPINS	packageInsDate	8	Package Install Date yyyymmdd	No	003
PRDMPFUN	promoFunction	8	PROMOTE vs DEMOTE	No	004
PRDMPTYP	promoType	6	FULL vs SELECT	No	
PRDMPSCP	promoScope	8	CHECK vs SERVICE (relevant to service exits only)	No	
PRDMOPTN	optionRequested	1	Option chosen from panel	No	
PRDMSITE	promoSite	8	Target site	No	005
PRDMPNAM	promoName	8	Target promotion name	No	006
PRDMPLVL	promoLevel	2	Target promotion level	Yes	007
PRDMLPNM	lastPromoName	8	Last promotion name	No	008
PRDMLPLV	lastPromoLevel	2	Last promotion level	No	009
PRDMPDTE	promoDate	10	Promotion date yyyy/mm/dd	No	010
PRDMPTME	promoTime	8	Promotion time hh:mm:ss	No	011
PRDMPUSR	promoUser	8	Promotion userid	No	012
PRDMSCHD	scheduleDate	8	Schedule date yyyymmdd	Yes	013
PRDMSCHT	scheduleTime	4	Schedule time hhmm	Yes	014
PRDMSLST	shortList	1	Short selection list Y/N	Yes	015
PRDMBYPO	bypassOverlayCheck	1	Bypass overlay check Y/N	Yes	016

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
PRDMSUPN	suppressNotify	1	Suppress batch messages Y/N	Yes	017
PRDMMIXC	mixedCase	1	Name has mixed case?	Yes	034
PRDMJOB1	jobCard01	72	Job card line #1	Yes	018
PRDMJOB2	jobCard02	72	Job card line #2	Yes	019
PRDMJOB3	jobCard03	72	Job card line #3	Yes	020
PRDMJOB4	jobCard04	72	job card line #4	Yes	021
PRDMFORC	demoteRequired	1	Prior demote required Y/N	No	022
PRDMUVPN	userVarPanel	8	User variable panel name	Yes	
PRDMUV01 - 05	userVariable01 userVariable02 userVariable03 userVariable04 userVariable05	8*5	Set of five 8-byte package user variables	Yes	024 025 026 027 028
PRDMUV06 - 10	userVariable06 userVariable07 userVariable08 userVariable09 userVariable10	72*5	Set of five 72-byte package user variables	Yes	029 030 031 032 033
PRDMUVAR	userVariables	1	Display User variable panel (Y/N)	Yes	023
PRDMOPRF	optsProfile	8	Used to select the ZDDOPTS profile for the display of user options for the ZMF Client Pack	Yes	
Repeated Group (Variable Length)					
PRDMCTYP	componentType.	3	Selected component libtype (stem variable, componentType.0 has number of instances)	No	
PRDMCOMP	componentName.	0-256	Selected component name (stem variable, componentName.0 has number of instances)	No	
Returned by Exit					
PRDMGO	proceed	3	Set to 'NO' to stop the file tailoring process	Yes	
PRDMLOKD	dataLocked	3	Fields locked? (YES/NO)	Yes	
PRDMSHRT	shortMsg	24	Short error message text	Yes	
PRDMLONG	longMsg	128	Long error message text	Yes	

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
PRDMCURS	cursorField	3	For ISPF where you wish the cursor to be placed on return to the panel display. The values relating to each field are shown in this table.	Yes	
PRDMCHNG	dataChanged	3	This field must be set to YES if you wish to return changed values to ZMF.	Yes	

Audit

This section describes the audit functional area of the high-level language exits. The 4-character exit name identifier is AUDT.

Select option 8 Audit from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define exits for the audit job submission and audit process:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> _____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed. Here is how a sample panel might look:

```
CMNHLLMN                      HLL Exit Definition            Row 1 to 6 of 9
Command ===>                                                 Scroll ===> CSR

Internal External  + Active  1=LE    Description  +
  Name     Name              2=REXX  Debug Userids  +
-------------------------------------------------------------------------------
AUDT00AR              NO     2       pre  autoresolve job submission
 Debug:               NO     2
-------------------------------------------------------------------------------
AUDT01JB              NO     2       post all audit job processing
 Debug:               NO     2
-------------------------------------------------------------------------------
AUDT00RC              NO     2       pre  audit job setting of package RC
 Debug:               NO     2
-------------------------------------------------------------------------------
AUDT00XM              NO     2       pre  audit job submission service
 Debug:               NO     2
-------------------------------------------------------------------------------
AUDT01XM              NO     2       post audit job submission service
 Debug:               NO     2
-------------------------------------------------------------------------------
AUDT0001              NO     2       pre  audit submission panel
 Debug:               NO     2
...
```

The audit functional area is a hybrid in that there are the usual panel user-interface exit points around the audit submission process; but, exit points can also be taken from within the audit batch process itself. These exit points allow the administrator to execute external processes from audit results or to alter the return code that is being set for each

package. As the environment in which these two different classes of exits execute is quite different, the data supplied to them also varies.

The panels around which the audit exit points can be placed are:

Panel ID	Description	Exit Name
CMNAUDIT	Audit submission main panel	AUDT0001 AUDT0101
CMNAUDAP	Applications-in-scope selection panel	AUDT0002 AUDT0102

The audit job submission services exits are:

Exit Name	Description
AUDT00XM	Audit job submission service pre-exit
AUDT01XM	Audit job submission service post-exit

The batch job audit exits are:

Exit Name	Description
AUDT00AR	Exit that is taken prior to the submission of each autoresolve job.
AUDT00RC	Exit that is taken prior to setting the audit return code for each package.
AUDT01JB	Exit that is taken at the end of the audit job.

Sample exits are provided in the CMNZMF.SAMPLES distribution library. These examples show how to list all the information incoming to these exits. Note that not all information is available to all exit points. The exits that occur early in the dialog will not have as much information as the exits that occur later in the dialog.

The audit exit examples are:

CMNZMF.SAMPLES Library Member	Description
HXCAUDT	COBOL example
HXPAUDT	PL/I example
HXRAUDT	REXX example
HXRASCP	REXX example of how to set extra applications-in-scope

The majority of the fields are fixed length. (See the data interface below.) However, there are a couple of optional variable length sections. When filled in, these sections contain information about the applications-in-scope for this audit and the participating packages included in the audit.

The format for this variable length data in COBOL is:

```
***
* VARIABLE BLOCK POINTER
*
* EACH ENTRY IN A VARIABLE LENGTH BLOCK CONSISTS OF THE DATA
* FOLLOWED BY A POINTER TO THE NEXT ENTRY. WHEN THAT POINTER
* IS NULL THEN THERE ARE NO FURTHER ENTRIES IN THE BLOCK.
***
     03 AUDTVB1L-PTR        USAGE IS POINTER.
     03 AUDTVB2L-PTR        USAGE IS POINTER.
*
***
* APPLICATIONS IN SCOPE
***
 01  AUDTVB1.
     03 AUDTASCP            PIC X(4).
*                                       APPLICATION
     03  PTR-NEXT-AUDTVB1   POINTER.
*                                       POINTER TO NEXT ENTRY
***
* ELEGIBLE PARTICIPATING PACKAGES
***
 01  AUDTVB2.
     03 AUDTPPKG            PIC X(10).
*                                       PPKG NAME
     03 AUDTPPLV            PIC X(1).
*                                       PPKG LEVEL
     03 AUDTPPTY            PIC X(1).
*                                       PPKG TYPE
     03 AUDTPPST            PIC X(3).
*                                       PPKG STATUS
     03 AUDTPPDP            PIC X(4).
*                                       PPKG DEPARTMENT
     03 AUDTPPIN            PIC X(8).
*                                       PPKG INSTALL DATE YYYYMMDD
     03  PTR-NEXT-AUDTVB2   POINTER.
*                                       POINTER TO NEXT ENTRY
```

The format for this variable length data in PL/I is:

```
/***                                                              */
 /* VARIABLE BLOCK POINTERS                                       */
 /* EACH POINTS TO A DIFFERENT VARIABLE LENGTH SECTION OF DATA.   */
 /* SECTIONS MAY BE MISSING DEPENDING ON FUNCTION AND PACKAGE     */
 /* TYPE. IF THEY ARE MISSING THEN THE POINTER WILL BE NULL.      */
 /*                                                               */
 /* EACH ENTRY IN A VARIABLE LENGTH BLOCK CONSISTS OF THE DATA    */
 /* FOLLOWED BY A POINTER TO THE NEXT ENTRY. WHEN THAT POINTER    */
 /* IS NULL THEN THERE ARE NO FURTHER ENTRIES IN THE BLOCK.       */
 /***                                                           */
         2 AUDTVB1P               PTR,
         2 AUDTVB2P               PTR;
 /***                                                           */
 /* APPLICATIONS IN SCOPE LIST                                    */
 /***                                                           */
 DCL     1 AUDTVB1                BASED(WORKVB1P),
              2 AUDTASCP          CHAR(4),  /*APPLICATION         */
              2 PTR_NEXT_AUDTVB1  PTR;      /*PTR TO NEXT ENTRY   */
 /***                                                           */
 /* ELIGIBLE PARTICIPATING PACKAGE LIST                           */
 /***                                                           */
 DCL     1 AUDTVB2                BASED(WORKVB2P),
              2 AUDTPPKG          CHAR(10), /*PPKG NAME          */
              2 AUDTPPLV          CHAR(1),  /*PPKG LEVEL         */
              2 AUDTPPTY          CHAR(1),  /*PPKG TYPE          */
              2 AUDTPPST          CHAR(3),  /*PPKG STATUS        */
              2 AUDTPPDP          CHAR(4),  /*PPKG DEPARTMENT    */
              2 AUDTPPIN          CHAR(8),  /*PPKG INSTALL DATE  */
              2 PTR_NEXT_AUDTVB2  PTR;      /*PTR TO NEXT ENTRY   */
```

The method for traversing this variable length list is the same as that used in other functions. The anchor pointers (AUDTVB1P and AUDTVB2P) point to the first in the chain of entries (null if no chain exists). Each entry contains a pointer to the next entry (null at end of chain).

REXX makes use of stem variables as usual with a variable number of similar data items.

The availability of the information in these variable sections is:

Exit Name	Participating Package Information Available? (Modifiable?)	Applications-in-Scope Information Available? (Modifiable?)
AUDT0001	NO (NO)	NO (NO)
AUDT0101	NO (NO)	YES (YES)
AUDT0002	NO (NO)	YES (YES)
AUDT0102	NO (NO)	YES (YES)
AUDT00XM	NO (NO)	YES (YES)
AUDT01XM	NO (NO)	YES (NO)
AUDT00RC	YES (NO)	NO (NO)
AUDT01JB	YES (NO)	NO (NO)
AUDT00AR	YES (NO)	NO (NO)

A single data structure is passed to all of these exits. The data interface looks like this:

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
AUDTFUNC	function	8	Internal exit name	No	
AUDTDBUG	debugCall	1	Debug exit call (Y/N)	No	
AUDTORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL	No	
AUDTZMFS	zmfSubs	1	ZMF subsystem character	No	
AUDTPDB2	db2Subs	4	Default Db2 subsystem for this ZMF	No	
AUDTUSER	userid	8	Userid for function calling this exit	No	
AUDTEXTN	externalName	256	External routine name defined for this exit	No	
AUDTPKGN	packageId	10	The package being acted on	No	001
AUDTSCOP	auditScope	1	Display appls in scope Y/N	Yes	002
AUDTMODE	auditMode	1	Staging libs only Y/N	Yes	003
AUDTINCH	includeHistory	1	Include history Y/N	Yes	004
AUDTFMTR	printFormat	1	Format for printing Y/N	Yes	005
AUDTAPSP	auditAsSimple	1	Audit as simple Y/N	Yes	006
AUDTAPPP	auditAsPrimary	1	Audit as primary Y/N	Yes	006
AUDTAPDP	auditByDept	1	Audit by department Y/N	Yes	006
AUDTTRUT	restrictRcToTarget	1	Update target pkg RC only Y/N	Yes	007
AUDTTRCO	auditWithTrace	1	Audit trace Y/N	Yes	
AUDTXHDR	includeXapHeaders	1	Show XAP headers T/Y/N	Yes	008
AUDTSUPM	suppressNotify	1	Suppress batch messages Y/N	Yes	009
AUDTLOCK	lockPackage	1	Lock package for audit Y/N	Yes	010
AUDTMIXC	mixedCase	1	Name has mixed case?	Yes	029
AUDTRPLK	resetPackageLock	1	Reset package lock for audit Y/N	Yes Note #1	028
AUDTAUTR	autoResolve	1	Autoresolve Y/N	Yes	011

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
AUDTAUTP	autoResolveParms	54	Autoresolve parameter dsn(member)	Yes	012
AUDTJOB1	jobCard01	72	Job card line #1	Yes	013
AUDTJOB2	jobCard02	72	Job card line #2	Yes	014
AUDTJOB3	jobCard03	72	Job card line #3	Yes	015
AUDTJOB4	jobCard04	72	Job card line #4	Yes	016
AUDTUV01-05	userVariable01 userVariable02 userVariable03 userVariable04 userVariable05	8 * 5	Set of five 8-byte package user variables	Yes	018 019 020 021 022
AUDTUV06-10	userVariable06 userVariable07 userVariable08 userVariable09 userVariable10	72 * 5	Set of five 72-byte package user variables	Yes	023 024 025 026 027
AUDTUVAR	userVariables	1	Display User variable panel (Y/N)	Yes	
AUDTOPRF	optsProfile	8	Used to select the ZDDOPTS profile for the display of user options for the ZMF Client Pack	Yes	
Supplied to AUDT00AR Only					
AUDTARTN	arsTargetName	256	Target component name	No	
AUDTAREV	arsEvent	2	Autoresolve event	No	
AUDTARSY	arsSynchError	2	Caused by this synch error	No	
AUDTARSL	arsSourceLibtype	3	Source component libtype	No	
AUDTARSN	arsSourceName	256	Source component name	No	
AUDTARTP	arsTargetPkg	10	Target package for re-build	No	
AUDTARTL	arsTargetLibtype	3	Target libtype (used by relinks)	No	
AUDTARCP	arsCausalPkg	10	Package containing error causing this autoresolve action	No	
AUDTARCR	arsReplaceCsect	16	Csect to be replaced (relink)	No	

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
Supplied to AUDT00RC Only					
AUDTRTCD	auditReturnCode	2	Package audit return code	Yes	
Supplied to AUDT01JB Only					
AUDTRDSN	reportDsn	44	Dataset name allocated to CMNAD500 ddname AUDITRPT	No Note #2	
AUDTRMBR	reportMember	8	Member name for same, or blank if dsname is sequential	No Note #2	
Repeated Group #1 (Variable Length)					
AUDTASCP	applInScope.*n*	4	Application	Note #3	
Repeated Group #2 (Variable Length)					
AUDTPPKG	ppkg.name.n	10	Participating package	No	
AUDTPPLV	ppkg.level.n	1	Level	No	
AUDTPPTY	ppkg.type.n	1	Type	No	
AUDTPPST	ppkg.status.n	3	Status	No	
AUDTPPDP	ppkg.dept.n	4	Department	No	
AUDTPPIN	ppkg.installDate.n	8	Install date	No	
Returned by Exit					
AUDTGO	proceed	3	Set to 'NO' to stop the file tailoring process	Yes	
AUDTLOKD	dataLocked	3	Fields locked? (YES/NO)	Yes	
AUDTSHRT	shortMsg	24	Short error message text	Yes	
AUDTLONG	longMsg	128	Long error message text	Yes	
AUDTCURS	cursorField	3	For ISPF where you wish the cursor to be placed on return to the panel display. The values relating to each field are shown in this table.	Yes	
AUDTCHNG	dataChanged	3	This field must be set to YES if you wish to return changed values to ZMF.	Yes	

Note #1	The ISPF client forces this value to be NO each time the panel is displayed via the)INIT section of the panel. This is done in order to avoid users accidentally unlocking packages.
Note #2	The dsname,and, if stored in a PDS/E, the member name of the file to which the AUDITRPT ddname in the CMNAD500 step is allocated will be passed in these fields.
Note #3	A list of applications selected as being in scope are supplied on the call to the exit. Where relevant, the exit may add to, or remove from, this list. If you change the number of entries, ensure that the pointer chains are set up correctly (LE), or ensure the new total number of entries is assigned to applInScope.0 (REXX). If you return an empty list, *no* updates are made to the applications-in-scope list used by the audit submission function.

Freeze, Unfreeze, and Refreeze

This section describes the freeze functional area of the high-level language exits. The 4-character exit name identifier is FREZ.

Select option 9 Freeze from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define exits for the package freeze and selective unfreeze/refreeze processes:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed. Here is how a sample panel might look:

```
CMNHLLMN                      HLL Exit Definition           Row 1 to 13 of 13
Command ===> ____________________________________________________ Scroll ===> CSR

Internal External  + Active  1=LE    Description  +
  Name     Name              2=REXX  Debug Userids  +
-------------------------------------------------------------------------------
FREZ00UF _________      NO    2       pre package freeze/unfreeze panel
 Debug:  _________      NO    2       ________________________________________
-------------------------------------------------------------------------------
FREZ01UF _________      NO    2       post package freeze/unfreeze panel
 Debug:  TST#FREZ       NO    2       USER12 ,USER13
-------------------------------------------------------------------------------
FREZ01U1 _________      NO    2       selective component freeze/unfreeze
 Debug:  TST#FREZ       NO    2       USER12 ,USER13
-------------------------------------------------------------------------------
FREZ01U2 _________      NO    2       selective utility freeze/unfreeze
 Debug:  _________      NO    2       ________________________________________
-------------------------------------------------------------------------------
FREZ00XM _________      NO    2       pre  package freeze service
 Debug:  _________      NO    2       ________________________________________
-------------------------------------------------------------------------------
FREZ01XM _________      NO    2       post package freeze service
 Debug:  _________      NO    2       ________________________________________
-------------------------------------------------------------------------------
FREZ00XR _________      NO    2       pre  selective refreeze service
 Debug:  _________      NO    2       ________________________________________
-------------------------------------------------------------------------------
FREZ01XR _________      NO    2       post selective refreeze service
 Debug:  _________      NO    2       ________________________________________
-------------------------------------------------------------------------------
FREZ00XU _________      NO    2       pre  selective unfreeze service
 Debug:  _________      NO    2       ________________________________________
-------------------------------------------------------------------------------
FREZ01XU _________      NO    2       post selective unfreeze service
 Debug:  _________      NO    2       ________________________________________
-------------------------------------------------------------------------------
FREZ0101 _________      NO    2       package freeze submenu
 Debug:  TST#FREZ       NO    2       USER12 ,USER13
-------------------------------------------------------------------------------
FREZ0002 _________      NO    2       pre batch freeze submit panel
 Debug:  _________      NO    2       ________________________________________
-------------------------------------------------------------------------------
FREZ0102 _________      NO    2       post batch freeze submit panel
 Debug:  _________      NO    2       ________________________________________
****************************** Bottom of data ******************************
```

The panels around which exit points will be placed are listed below. The internal exit name (also known as function code) is FREZ0*pnn*, where:

- *p*=0 is the pre-exit.
- *p*=1 is the post-exit.
- *nn* is an alphanumeric identifier relating to the panel for which the exit is taken.

The pre-exit is taken before the panel is displayed and the post-exit is taken after the panel has been displayed.

An internal exit name of FREZ0*p*01, for example, means that both pre- and post-exits exist. That is, the name of the pre-exit is FREZ0002 and the name of the post-exit is FREZ0102. If it makes no sense to have a pre-exit, the internal exit name is given as

FREZ0101 (post-exit only). If it makes no sense to have a post-exit, the internal exit name is given as FREZ0001 (pre-exit only).

Most table displays have only post-exits. That is, we do not want to have a pre-exit that manipulates the lists that ZMF generates. We may want to have a post-exit to validate the selections that the user makes from the lists.

The panels around which the freeze, unfreeze, or refreeze exit points are placed are:

Panel Id	Description	Exit Name
CMNFRZ01	Package freeze submenu	FREZ0101
CMNFRZ02	Batch package freeze submit panel	FREZ0002 FREZ0102
CMNUNFRZ	Package unfreeze/refreeze	FREZ00UF FREZ01UF
CMNUNF01	Selective component unfreeze/refreeze	FREZ01U1
CMNUNF02	Selective utility request unfreeze/refreeze	FREZ01U2

Pre- and post-XML-service calls for freeze, unfreeze, and refreeze are:

XML Service Name	Description	Exit Name
package.service.freeze	Pre-service call for full package freeze	FREZ00XM
package.service.freeze	Post-service call for full package freeze	FREZ01XM
package.src_lod.unfreeze	Pre-service call for selective component unfreeze	FREZ00XU
package.src_lod.unfreeze	Post-service call for selective component unfreeze	FREZ01XU
package.src_lod.refreeze	Pre-service call for selective component refreeze	FREZ00XR
package.src_lod.refreeze	Post-service call for selective component refreeze	FREZ01XR

Sample exits are provided in the CMNZMF.SAMPLES distribution library. These examples show how to list all the information incoming to these exits. Note that not all information is available to all exit points. The exits that occur early in the dialog will not have as much information as the exits that occur later in the dialog.

The client-driven selective unfreeze/refreeze post-exits (that is, after the components to be actioned are selected) are driven once per component with single-valued component information supplied on each call. The unfreeze and refreeze service-driven exits are only called once and are supplied with a list of selected component names and library types (with no other specific component information).

The freeze exit examples are:

CMNZMF.SAMPLES Library Member	Description
HXCFREZ	COBOL example
HXPFREZ	PL/I example
HXRFREZ	REXX example

A single data structure is passed to all of these exits. The data interface looks like this:

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
FREZFUNC	function	8	Internal exit name	No	
FREZDBUG	debugCall	1	Debug exit call (Y/N)	No	
FREZORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL	No	
FREZZMFS	zmfSubs	1	ZMF subsystem character	No	
FREZPDB2	db2Subs	4	Default Db2 subsystem for this ZMF	No	
FREZUSER	userid	8	Userid for function calling this exit	No	
FREZEXTN	externalName	256	External routine name defined for this exit	No	
FREZACTN	freezeAction	1	F/U/R/S	Yes	001
FREZOPTN	optionRequested	1	1/2/3/4/5	Yes	002
FREZPKGN	packageId	10	The package being acted on	No	
FREZPCAT	packageCategory	8	Same as service scope value	No	
FREZCAST	categoryStatus	8	Frozen/unfrozen	No	
FREZCSTD	Component staged date	8	Selective post exits only	No	
FREZCSTT	Component staged time	6	Selective post exits only	No	
FREZCUSR	Component user	8	Selective post exits only	No	
FREZCSTA	Component status	8	(frozen/unfrozen) selective only	No	
FREZUREQ	utilityRequest	3	Scr/ren (scratch/rename)	No	

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
FREZNWNM	componentNewName	256	For rename request	No	
FREZJOB1	jobCard01	72	Job card line #1	Yes	013
FREZJOB2	jobCard02	72	Job card line #2	Yes	014
FREZJOB3	jobCard03	72	Job card line #3	Yes	015
FREZJOB4	jobCard04	72	Job card line #4	Yes	016
FREZMIXC	mixedCase	1	Name has mixed case?	Yes	028
FREZVAL0	validateOnly	1	Validate the package for freeze only (FREZ0xXM only)	No	
FREZUVPN	userVarPanel	8	User variable panel name	Yes	
FREZTUV01 - 05	userVariable01 userVariable02 userVariable03 userVariable04 userVariable05	8 * 5	Set of five 8-byte package user variables	Yes	018 019 020 021 022
FREZUV06 - 10	userVariable06 userVariable07 userVariable08 userVariable09 userVariable10	72 * 5	Set of five 72-byte package user variables	No	023 024 025 026 027
FREZMCNT	memberCount	5	Count of following member names & types	No	
FREZCTYP	componentType.0	3	Selected Component type. Stem variable	No	
FREZCOMP	componentName.0	1-256	Selected Component name (variable length). Stem variable	No	
FREZNOGO	proceed	3	Set to 'NO' to stop the process	Yes	
FREZLOKD	dataLocked	3	Fields locked? (YES/NO)	Yes	
FREZSHRT	shortMsg	24	Short error message text	Yes	
FREZLONG	longMsg	128	Long error message text	Yes	
FREZCURS	cursorField	3	For ISPF where you wish the cursor to be placed on return to the panel display. The values relating to each field are shown in this table.	Yes	

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
FREZCHNG	dataChanged	3	This field must be set to YES if you wish to return changed values to ZMF.	Yes	
FREZUVAR	userVariables	1	Display User variable panel (Y/N)	Yes	017

Package Approve and Reject

This section describes the package approve/reject functional area of the high-level language exits. The 4-character exit name identifier is APRV.

Select option A Approve/Reject from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define exits for the package approve and reject functions:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed. Here is how a sample panel might look:

```
CMNHLLMN                      HLL Exit Definition            Row 1 to 10 of 11
Command ===> ____________________________________________________ Scroll ===> CSR

Internal External  + Active  1=LE    Description  +
  Name     Name              2=REXX  Debug Userids  +
-------------------------------------------------------------------------------
APRV00XM _________      NO     2       pre  service all for approve
 Debug:  HXRAPRV        NO     2       USER12
-------------------------------------------------------------------------------
APRV01XM _________      NO     2       post service call for approve
 Debug:  HXRAPRV        NO     2       USER12
-------------------------------------------------------------------------------
APRV0101 _________      NO     2       list of packages to be approved
 Debug:  HXRAPRV        YES    2       USER12
-------------------------------------------------------------------------------
APRV0102 _________      NO     2       approve/reject option menu
 Debug:  HXRAPRV        YES    2       USER12
-------------------------------------------------------------------------------
APRV0103 _________      NO     2       approver entity list
 Debug:  APRV0103       YES    2       USER12
-------------------------------------------------------------------------------
APRV0004 _________      NO     2       pre  checkoff comments
 Debug:  HXRAPRV        YES    2       USER12 ,USER58
-------------------------------------------------------------------------------
APRV0104 _________      NO     2       post checkoff comments
 Debug:  HXRAPRV        YES    2       USER12 ,USER58
-------------------------------------------------------------------------------
APRV0105 _________      NO     2       reject reasons entity selection
 Debug:  HXRAPRV        YES    2       USER12
-------------------------------------------------------------------------------
APRV0006 _________      NO     2       pre  reject reasons text
 Debug:  HXRAPRV        YES    2       USER12 ,USER58
-------------------------------------------------------------------------------
APRV0106 _________      NO     2       post reject reasons text
 Debug:  HXRAPRV        YES    2       USER12 ,USER58
...
```

The panels around which exit points will be placed are listed below. The internal exit name (also known as function code) is APRV0*pnn*, where:

- *p*=0 is the pre-exit.
- *p*=1 is the post-exit.
- *nn* is an alphanumeric identifier relating to the panel for which the exit is taken.

The pre-exit is taken before the panel is displayed. The post-exit is taken after the panel has been displayed.

An internal exit name of APRV0*p*04, for example, means that both pre- and post-exits exist. That is, the name of the pre-exit is APRV0004 and the name of the post-exit is APRV0104. If it makes no sense to have a pre-exit, the internal name will be given as APRV0101 (post-exit only). If it makes no sense to have a post-exit, the internal exit name is given as APRV0001 (pre-exit only).

Most table displays have only post-exits. That is, we do not want to have a pre-exit that manipulates the lists that ZMF generates. We may want to have a post-exit to validate the selections that the user makes from the lists.

The panels around which the approve/reject exit points are placed are:

Panel Id	Description	Exit Name
CMNAPPL1	Resulting list of packages to be approved (short)	APRV0101
CMNAPPL2	Resulting list of packages to be approved (long)	APRV0101
CMNAPPOP	Approve function submenu	APRV0102
CMNAPPLS	Approver entity list for action	APRV0103
CMNCHKLS	Checkoff comments	APRV0004 APRV0104
CMNREJR0	Reject reason entity selection	APRV0105
CMNREJR1	Reject reason text	APRV0006 APRV0106

Pre- and post-XML-service calls for approve are:

XML Service Name	Description	Exit Name
package.service.approve	Pre-service call for approve (all actions)	APRV00XM
package.service.approve	Post-service call for approve	APRV01XM

Sample exits are provided in the CMNZMF.SAMPLES distribution library. These examples h show how to list all the information coming in to the exits. Not all information is available to all exits. The exits that occur early in the dialog will not have as much information as the exits that occur later in the dialog.

The approve exit examples are:

CMNZMF.SAMPLES Library Member	Description
HXCAPRV	COBOL example
HXPAPRV	PL/I example
HXRAPRV	REXX example
HXRAPRV3	REXX example exit to warn the approver if check has found an issue and to give the approver two options: ■ Ignore warnings and proceed. ■ Cancel approval.

A single data structure is passed to all of these exits. The data interface looks like this:

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
APRVFUNC	function	8	Internal exit name	No	
APRVDBUG	debugCall	1	Debug exit call (Y/N)	No	

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
APRVORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL	No	
APRVZMFS	zmfSubs	1	ZMF subsystem character	No	
APRVPDB2	db2Subs	4	Default Db2 subsystem for this ZMF	No	
APRVUSER	userid	8	Userid for function calling this exit	No	
APRVEXTN	externalName	256	External routine name defined for this exit	No	
APRVACTN	approverAction	1	(approve/reject/review /checkoff)	Yes	001
APRVPKGN	packageId	10	The package being acted on	No	
APRVPSTA	packageStatus	1	See service values	No	
APRVPINS	packageInstallDate	8	Package installation date	No	
APRVPLVL	packageLevel	1	Package level	No	
APRVPTYP	packageType	1	Package type	No	
APRVDEPT	packageDepartment	4	Department	Yes	
APRVWKRQ	packageWorkRequest	12	Work request number	Yes	
APRVCRTR	packageCreator	8	User id of the package creator	No	
APRVPRST	promotionSite	8	Promotion site	No	
APRVPRNM	promotionName	8	Promotion name	No	
APRVPRLV	promotionLevel	2	Promotion level	No	
APRVPRDT	promotionDate	8	Promotion date	No	
APRVPRTM	promotionTime	6	Promotion time	No	
APRVPRUS	promotionUser	8	Promotion userid	No	
APRVAENT	approvalEntity	8	Approval entity	No	
APRVADSC	approvalDescription	44	Approval description	No	
APRVACDE	approvalCode	1	Approval code	No	
APRVAUSR	approvalUser	8	Userid of the approver	No	
APRVADTE	approvalDate	8	Approval date	No	
APRVATME	approvalTime	6	Approval time	No	
APRVAORD	orderNumber	2	Order number	No	

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
APRVAFUN	lastFunction	7	Last function	No	
APRVREJR	reasons.*n*	72	Stem variable. 10 reject reasons	Yes	010-019
APRVCKCM	comments.*n*	72	Stem variable 14 checkoff comments	Yes	031-044
APRVGO	proceed	3	Set to 'NO' to stop the process	Yes	
APRVLOKD	dataLocked	3	Fields locked? (YES/NO)	Yes	
APRVSHRT	shortMsg	24	Short error message text	Yes	
APRVLONG	longMsg	128	Long error message text	Yes	
APRVCURS	cursorField	3	For ISPF where you wish the cursor to be placed on return to the panel display. The values relating to each field are shown in this table.	Yes	
APRVCHNG	dataChanged	3	This field must be set to YES if you wish to return changed values to ZMF.	Yes	

Revert/Backout

This section describes the revert and backout functional areas of the high-level language exits. The 4-character exit name identifier is RVRT.

Select option B Revert/Backout from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define exits for the package revert and backout functions:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslist service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed. Here is how the default unmodified panel looks:

```
CMNHLLMN                      HLL Exit Definition           Row 1 to 16 of 16
Command ===> ____________________________________________________ Scroll ===> CSR

Internal External  + Active  1=LE    Description  +
  Name     Name              2=REXX  Debug Userids  +
-------------------------------------------------------------------------------
RVRT01B1 _________      NO     2       package backout selection panel
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT00B2 _________      NO     2       pre  backout reason entry panel
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT01B2 _________      NO     2       post backout reason entry panel
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT01B3 _________      NO     2       backout site selection
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT00B4 _________      NO     2       pre  backout remote submission
 Debug:  _________      NO     2
-------------------------------------------------------------------------------
RVRT01B4 _________      NO     2       post backout remote submission
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT00XB _________      NO     2       pre  service call for backout
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT01XB _________      NO     2       post service call for backout
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT00XM _________      NO     2       pre  service call for revert
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT01XM _________      NO     2       post service call for revert
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT0101 _________      NO     2       package revert selection panel
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT0002 _________      NO     2       pre  revert reason entry panel
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT0102 _________      NO     2       post revert reason entry panel
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT0103 _________      NO     2       revert site selection
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT0004 _________      NO     2       pre  revert remote submission
 Debug:  _________      NO     2       ____________________________________
-------------------------------------------------------------------------------
RVRT0104 _________      NO     2       post revert remote submission
 Debug:  _________      NO     2       ____________________________________
****************************** Bottom of data ********************************
```

The panels around which exit points will be placed are listed below. The internal exit name (also know as function code) is RVRT0*pnn*, where:

- *p*=0 is the pre-exit
- *p*=1 is the post-exit

- *nn* is an alphanumeric identifier relating to the panel for which the exit is taken. If the first n is B then it is a backout exit, a 0 a revert exit, or if nn is XM then it is a revert pre or post service call, and XB it is a backout pre or post service call.

The pre-exit is taken before the panel is displayed. The post-exit is taken after the panel has been displayed.

An internal exit name of RVRT0*p*04, for example, means that both pre- and post-exits exist. That is, the name of the pre-exit is RVRT0004 and the name of the post-exit is RVRT0104. If it makes no sense to have a pre-exit, the internal name is given as RVRT0101 (post-exit only). If it makes no sense to have a post-exit, the internal name is given as RVRT0001 (pre-exit only).

Most table displays have only post-exits. That is, we do not want to have a pre-exit that manipulates the lists that ZMF generates. We may want to have a post-exit to validate the selections that the user makes from the lists.

The panels around which the revert exit points are placed are:

Panel Id	Description	Exit Name
CMNREV00	Package revert main panel	RVRT0101
CMNREVRS	Revert reason entry panel	RVRT0002 RVRT0102
CMNRVSTI	Revert site selection	RVRT0103
CMNRVJCD	Revert remote submission panel	RVRT0004 RVRT0104

The panels around which the backout exit points are placed are:

Panel Id	Description	Exit Name
CMNBKOUT	Post main backout entry panel	RVRT01B1
CMNBKRSN	Pre backout reason panel	RVRT00B2
CMNBKRSN	Post backout reason panel	RVRT01B2
CMNBKSTI	Post backout site selection panel	RVRT01B3
CMNBKJCD	Pre backout jobcard specification panel	RVRT00B4
CMNBKJCD	Post backout jobcard specification panel	RVRT01B4

Pre- and post-XML-service calls for revert are:

XML Service Name	Description	Exit Name
package.service.revert	Pre-service call for package revert	RVRT00XM
package.service.revert	Post-service call for package revert	RVRT01XM

Pre- and post-XML-service calls for backout are:

XML Service Name	Description	Exit Name
package.service.backout	Pre-service call for package backout	RVRT00XB
package.service.backout	Post-service call for package backout	RVRT01XB

Sample exits are provided in the CMNZMF.SAMPLES distribution library. These examples show how to list all the information coming in to the exits. Not all information is available to all exits. The exits that occur early in the dialog will not have as much information as the exits that occur later in the dialog.

The revert and backout exit examples are:

CMNZMF.SAMPLES Library Member	Description
HXCRVRT	COBOL example
HXPRVRT	PL/I example
HXRRVRT	REXX example

A single data structure is passed to all of these exits. The data interface looks like this:

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
RVRTFUNC	function	8	Internal exit name	No	
RVRTDBUG	debugCall	1	Debug exit call (Y/N)	No	
RVRTORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL	No	
RVRTZMFS	zmfSubs	1	ZMF subsystem character	No	
RVRTPDB2	db2Subs	4	Default Db2 subsystem for this ZMF	No	
RVRTUSER	userid	8	Userid for function calling this exit	No	
RVRTEXTN	externalName	256	External routine name defined for this exit	No	
RVRTPKGN	packageId	10	The package being acted on	No	
RVRTPSTA	packageStatus	3	Package/Site status	No	
RVRTSITE	siteName	8	Install site	No	
RVRTINDT	pacakgeInsDate	8	Yyyymmdd	No	
RVRTFINT	fromInstallTime	6	Hhmm00	No	
RVRTTINT	toInstallTime	6	Hhmm00	No	

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
RVRTOANM	contactName	25	Orig. analyst name	No	
RVRTOAPH	contactPhone	15	Orig. analyst phone	No	
RVRTAANM	alternateContactName	25	Alt. analyst name	No	
RVRTAAPH	alternateContactPhone	15	Alt. analyst phone	No	
RVRTJOB1	jobCard01	72	Job card line #1	Yes	011
RVRTJOB2	jobCard02	72	Job card line #2	Yes	012
RVRTJOB3	jobCard03	72	Job card line #3	Yes	013
RVRTJOB4	jobCard04	72	Job card line #4	Yes	014
RVRTREVR	reasons.*n*	72	Set of nine 72-byte variables	Yes	021-029
RVRTGO	proceed	3	Set to 'NO' to stop the process	Yes	
RVRTLOKD	dataLocked	3	Fields locked? (YES/NO)	Yes	
RVRTSHRT	shortMsg	24	Short error message text	Yes	
RVRTLONG	longMsg	128	Long error message text	Yes	
RVRTCURS	cursorField	3	For ISPF where you wish the cursor to be placed on return to the panel display. The values relating to each field are shown in this table.	Yes	
RVRTCHNG	dataChanged	3	This field must be set to YES if you wish to return changed values to ZMF.	Yes	

Package Syslib

This section describes the Package Syslib functional area of the high-level language exits. The 4-character exit name identifier is SYSL.

The main exit point in this functional area is SYSL00XM, which is taken from the package.list.syslib service. For convenience, further exit points in the process of extracting a standard language identifier for a component are included in this functional area and are described after SYSL00XM below.

The SYSL00XM exit may be used to alter the default syslib concatenation generated by ZMF, or it may be used to prevent any result being returned to the service caller (a user specified message can be delivered in this case).

Sample exits are provided which show how to list all the information incoming to the exit.

The majority of the fields are fixed in nature (see list later in this document). However, there are three variable length lists which are passed to/from the exit. The first is a complete list of library types which are physically allocated to the package for which the service has been invoked.

The second is the list of syslib names (MVS libraries or zFS directories) as generated by the default ZMF logic (i.e. staging libraries, promotion libraries, baseline libraries).

The third list is null when the exit is called and is where the exit may place any altered list of syslib names. Details on how to generate an altered list of syslib names are provided in the sample exits (HXCSYSL - Cobol, HXPSYSL - PL/I, HXRSYSL - REXX). The format of this variable length data is given here.

In COBOL:

```
      ***
           * VARIABLE LENGTH LISTS FOLLOW, THERE ARE THREE.
           * IF THE LIST IS EMPTY THEN THE RELEVANT POINTER WILL BE NULL.
           * ELSE IT WILL POINT TO A REPEATING GROUP IN WHICH EACH ENTRY
           * CONSISTS OF A POINTER TO THE NEXT ENTRY FOLLOWED BY THE DATA.
           * WHEN THE POINTER IS NULL THEN THERE NO FURTHER ENTRIES IN
           * THE BLOCK.
           *
           * 1) LIBTYPES ALLOCATED TO THE REQUESTED PACKAGE
           * 2) THE DEFAULT SYSLIB LIST PROVIDED BY THE ZMF SERVICE
           *
           * THE THIRD LIST, IF REQUIRED, WILL BE CREATED BY YOU (SAMPLE
           * CODE IS PROVIDED IN HXCSYSL). IF YOU WISH TO CHANGE THE SET
           * OF NAMES USED IN THIS SYSLIB CONCATENATION THEN YOU MUST
           * PROVIDE THE ALTERED LIST AS A LINKED LIST WITH THE FORMAT
           * EXACTLY AS REPRESENTED IN THIS COPYBOOK.
           * THE POINTER TO THE FIRST ITEM ON THE LIST IS PLACED AT
           * SYSLLST3-PTR AND THE POINTER TO THE NEXT ENTRY ON THE LIST
           * MUST BE NULL TO TERMINATE THE LIST.
           * STORAGE FOR THE LIST ENTRIES MUST BE TAKEN FROM LE HEAPID=0
           * USING THE CEEGTST FUNCTION.
           *
           * ON ENTRY TO THE EXIT SYSLLST3-PTR IS NULL. IF YOU DO NOT
           * INTEND TO PROVIDE A CHANGED LIST THEN IT IS IMPORTANT THAT
           * YOU LEAVE SYSLLST3-PTR AS NULL.
           *
           * TO HAVE ZMF TAKE NOTE OF YOUR CHANGED LIST YOU NEED TO SET
           * SYSLCHNG TO "YES".
           ***
                03 SYSLLST1-PTR        USAGE IS POINTER.
                03 SYSLLST2-PTR        USAGE IS POINTER.
                03 SYSLLST3-PTR        USAGE IS POINTER.
```

```
*
***
* VARIABLE LENGTH AREA FOR ALLOCATED STAGING LIBTYPE LIST
***
 01  SYSLLST1.
     03 PTR-NEXT-SYSLLST1   POINTER.
*                                    POINTER TO NEXT ENTRY
     03 SYSLLTPN            PIC X(3).
*                                    A SINGLE LIBTYPE ENTRY
***
* VARIABLE LENGTH AREA FOR THE SERVICE GENERATED SYSLIB LIST
***
 01  SYSLLST2.
     03 PTR-NEXT-SYSLLST2   POINTER.
*                                    POINTER TO NEXT ENTRY
     03 SYSLFROM            PIC X(1).
*                                    S,P,B - STG,PROMO,BASELINE
     03 SYSLATTR            PIC X(18).
*                                    FROM ATTRIBUTES
     03 SYSLSTGE            REDEFINES SYSLATTR.
*                                    STAGING ATTRIBUTES:FROM=S
        09 SYSLPKG          PIC X(10).
*                                    PACKAGE
        09 FILLER           PIC X(8).
*
     03 SYSLPROM            REDEFINES SYSLATTR.
*                                    PROMO ATTRIBUTES:  FROM=P
        09 SYSLPRST         PIC X(8).
*                                    PROMOTION SITE
        09 SYSLPRLV         PIC X(2).
*                                    PROMOTION LEVEL
        09 SYSLPRNM         PIC X(8).
*                                    PROMOTION NAME
     03 SYSLAPPL            PIC X(4).
*                                    APPLICATION
     03 SYSLLIBT            PIC X(3).
*                                    LIBTYPE
     03 SYSLSPEC            PIC X(1).
*                                    SPECIAL PROCESS IND.
     03 SYSLSUBT            PIC X(1).
*                                    SPECIAL PROCESS SUBTYPE
     03 SYSLDSNM.
        49 SYSLDSNM-LEN     PIC S9(4) COMP-5.
*                                              DSNAME LENGTH
        49 SYSLDSNM-VALUE   PIC X(1024).
*                                              DSNAME VALUE
***
* VARIABLE LENGTH AREA FOR SYSLIB LIST TO BE RETURNED TO ZMF
***
 01  SYSLLST3.
     03 PTR-NEXT-SYSLLST3   POINTER.
*                                    POINTER TO NEXT ENTRY
     03 SYSLIBNM.
        49 SYSLIBNM-LEN     PIC S9(4) COMP-5.
*                                              DSNAME LENGTH
        49 SYSLIBNM-VALUE   PIC X(1024).
*                                              DSNAME VALUE
```

And here in PL/I …

```
 /***                                                          */
      /* VARIABLE LENGTH LISTS FOLLOW, THERE ARE THREE.               */
      /* IF THE LIST IS EMPTY THEN THE RELEVANT POINTER WILL BE NULL.  */
      /* ELSE IT WILL POINT TO A REPEATING GROUP IN WHICH EACH ENTRY   */
      /* CONSISTS OF A POINTER TO THE NEXT ENTRY FOLLOWED BY THE DATA. */
      /* WHEN THE POINTER IS NULL THEN THERE NO FURTHER ENTRIES IN     */
      /* THE BLOCK.                                                    */
```

```
/*                                                                  */
/* 1) LIBTYPES ALLOCATED TO THE REQUESTED PACKAGE                   */
/* 2) THE DEFAULT SYSLIB LIST PROVIDED BY THE ZMF SERVICE           */
/*                                                                  */
/* THE THIRD LIST, IF REQUIRED, WILL BE CREATED BY YOU (SAMPLE      */
/* CODE IS PROVIDED IN HXPSYSL). IF YOU WISH TO CHANGE THE SET      */
/* OF NAMES USED IN THIS SYSLIB CONCATENATION THEN YOU MUST          */
/* PROVIDE THE ALTERED LIST AS A LINKED LIST WITH THE FORMAT        */
/* EXACTLY AS REPRESENTED IN THIS COPYBOOK.                         */
/* THE POINTER TO THE FIRST ITEM ON THE LIST IS PLACED AT           */
/* SYSLLST3-PTR AND THE POINTER TO THE NEXT ENTRY ON THE LIST       */
/* MUST BE NULL TO TERMINATE THE LIST.                              */
/* STORAGE FOR THE LIST ENTRIES MUST BE TAKEN FROM LE HEAPID=0      */
/* USING THE CEEGTST FUNCTION.                                      */
/*                                                                  */
/* ON ENTRY TO THE EXIT SYSLLST3-PTR IS NULL. IF YOU DO NOT         */
/* INTEND TO PROVIDE A CHANGED LIST THEN IT IS IMPORTANT THAT       */
/* YOU LEAVE SYSLLST3-PTR AS NULL.                                  */
/*                                                                  */
/* TO HAVE ZMF TAKE NOTE OF YOUR CHANGED LIST YOU NEED TO SET       */
/* SYSLCHNG TO "YES".                                               */
/***                                                               */
       2 SYSLLST1_PTR           PTR,
       2 SYSLLST2_PTR           PTR,
       2 SYSLLST3_PTR           PTR;
/***                                                               */
/* VARIABLE LENGTH AREA FOR ALLOCATED STAGING LIBTYPE LIST          */
/***                                                               */
  DCL 1 SYSLLST1             BASED(WORKPTR1),
       2 PTR_NEXT_SYSLLST1  PTR,        /*POINTER TO NEXT ENTRY */
       2 SYSLLTPN           CHAR(3);    /*LIBTYPE               */
/***                                                               */
/* VARIABLE LENGTH AREA FOR THE SERVICE GENERATED SYSLIB LIST       */
/* NOTE: 3 DIFFERENT STRUCTURES DEFINED HERE FOR THE MINOR          */
/*       DIFFERENCES BETWEEN STAGING, PROMOTION, AND BASELINE       */
/*       DERIVED ENTRIES                                            */
/***                                                               */
/***                                                               */
/* LIST 2 ENTRY DERIVED FROM STAGING                                */
/***                                                               */
  DCL 1 SYSLLST2_S            BASED(WORKPTR2),
       2 PTR_NEXT_SYSLLST2_S PTR,       /*POINTER TO NEXT ENTRY */
       2 SYSLFROM_S          CHAR(1),   /*=S FOR STAGING ENTRY  */
       2 SYSLATTR_PKG_S      CHAR(10),  /*PKG FOR THIS STG LIB  */
       2 SYSLATTR_FILLER_S   CHAR(8),
       2 SYSLAPPL_S          CHAR(4),   /*APPLICATION           */
       2 SYSLLIBT_S          CHAR(3),   /*LIBRARY TYPE          */
       2 SYSLSPEC_S          CHAR(1),   /*SPECIAL PROCESS IND   */
       2 SYSLSUBT_S          CHAR(1),   /*SPECIAL PROC SUBTYPE  */
       2 SYSLDSNM_S          CHAR(1024) VARYING;
                                        /*VARCHAR DSNAME        */
/***                                                               */
/* LIST 2 ENTRY DERIVED FROM PROMOTION                              */
/***                                                               */
  DCL 1 SYSLLST2_P            BASED(WORKPTR2),
       2 PTR_NEXT_SYSLLST2_P PTR,       /*POINTER TO NEXT ENTRY */
       2 SYSLFROM_P          CHAR(1),   /*=P FOR PROMOTION ENTRY*/
       2 SYSLATTR_SITE_P     CHAR(8),   /*PROMOTION SITE        */
       2 SYSLATTR_LEVEL_P    CHAR(2),   /*PROMOTION LEVEL       */
       2 SYSLATTR_NAME_P     CHAR(8),   /*PROMOTION NAME        */
       2 SYSLAPPL_P          CHAR(4),   /*APPLICATION           */
       2 SYSLLIBT_P          CHAR(3),   /*LIBRARY TYPE          */
       2 SYSLSPEC_P          CHAR(1),   /*SPECIAL PROCESS IND   */
       2 SYSLSUBT_P          CHAR(1),   /*SPECIAL PROC SUBTYPE  */
       2 SYSLDSNM_P          CHAR(1024) VARYING;
                                        /*VARCHAR DSNAME        */
/***                                                               */
```

```
/* LIST 2 ENTRY DERIVED FROM BASELINE                              */
/***                                                               */
  DCL 1 SYSLLST2_B            BASED(WORKPTR2),
        2 PTR_NEXT_SYSLLST2_B PTR,       /*POINTER TO NEXT ENTRY */
        2 SYSLFROM_B          CHAR(1),   /*=B FOR PROMOTION ENTRY*/
        2 SYSLATTR_FILLER_B   CHAR(18),
        2 SYSLAPPL_B          CHAR(4),   /*APPLICATION           */
        2 SYSLLIBT_B          CHAR(3),   /*LIBRARY TYPE          */
        2 SYSLSPEC_B          CHAR(1),   /*SPECIAL PROCESS IND   */
        2 SYSLSUBT_B          CHAR(1),   /*SPECIAL PROC SUBTYPE  */
        2 SYSLDSNM_B          CHAR(1024) VARYING;
                                         /*VARCHAR DSNAME        */
/***                                                               */
/* VARIABLE LENGTH AREA IN WHICH TO RETURN AN ALTERED LIST         */
/* SEE SAMPLE EXIT HXPSYSL FOR DETAILS ON HOW TO DO THIS.          */
/***                                                               */
  DCL 1 SYSLLST3              BASED(WORKPTR3),
        2 PTR_NEXT_SYSLLST3   PTR,       /*POINTER TO NEXT ENTRY */
        2 SYSLIBNM            CHAR(1024) VARYING;
                                         /*VARCHAR DSNAME        */
```

The method for traversing this variable length list is the same as that used in other functions. The 'anchor' pointer points to the first in the chain of entries (null if no chain exists). Each entry contains a pointer to the next entry (null at end of chain).

To return a modified syslib list you must generate a linked list, anchored at SYSLLST3-PTR, with the storage required for each list entry allocated from the LE heap with heaped=0. This is assumed by the ZMF function responsible for tidying up the list after it has been passed back.

In REXX we just make use of stem variables as is usual with a variable number of similar data items.

Sample exits are provided in the CMNZMF.SAMPLES distribution library. These examples show how to list all the information coming in to the exit.

The package syslib exit examples are:

CMNZMF.SAMPLES Library Member	Description
HXCSYSL	COBOL example
HXPSYSL	PL/I example
HXRSYSL	REXX example

Select option S Package Syslib from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define exits for the package syslib function:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed. Here is how a sample panel might look:

```
CMNHLLMN                      HLL Exit Definition              Row 1 to 1 of 1
Command ===> ____________________________________________ Scroll ===> CSR

Internal External  + Active  1=LE    Description  +
  Name     Name              2=REXX  Debug Userids  +
-------------------------------------------------------------------------------
SYSL00XM _______       NO    2       service call for package.list.syslib
 Debug:  _______       NO    2       ____________________________________
****************************** Bottom of data ******************************
```

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)
SYSLFUNC	function	8	Internal exit name	No
SYSLDBUG	debugCall	1	Debug exit call (Y/N)	No
SYSLORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL	No
SYSLZMFS	zmfSubs	1	ZMF subsystem character	No
SYSLPDB2	db2Subs	4	Default Db2 subsystem for this ZMF	No
SYSLUSER	userid	8	Userid for function calling this exit	No
SYSLEXTN	externalName	256	External routine name defined for this exit	No
SYSLPKGN	packageId	10	The package being acted on	No

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)
SYSLLTYP	componentType	3	Library type of component for which the syslib is being generated	No
SYSLPROC	buildProc	8	Procedure to be used to build the component	No
SYSLLANG	language	8	Language for the component	No
SYSLPPSM	processPpkgAsSimple	1	If the component package is participating then treat it as simple (cf. audit option)	No
SYSLPPID	processPpkgByInstallDate	1	Process participating packages by install date (cf. audit option)	No
SYSLPPDP	processPpkgByDepartment	1	Process participating packages by department (cf. audit option)	No
SYSLPLVL	packageLevel	1	1 – simple 2 – complex 3 – super 4 – participating	No
SYSLPTYP	packageType	1	1 – planned permanent 2 – planned temporary 3 – unplanned permanent 4 – unplanned temporary	No
SYSLCXPK	complexSuperPackage	10	If the component package is participating then this is its complex package	No
SYSLPSTA	packageStatus	3	Package status (DEV, FRZ, etc)	No
SYSLPINS	packageInsDate	8	Pkg Install Date yyyymmdd	No
SYSLCRDT	packageCreateDate	8	Date pkg was created yyyymmdd	No
SYSLCRTI	packageCreator	8	Userid under which pkg was created	No
SYSLDEPT	packageDepartment	4	Pkg department	No
SYSLWRQN	packageWorkRequest	12	Pkg work request number	No
SYSLRQNM	requestorName	25	Pkg requestors name	No
SYSLRQPH	requestorPhone	15	Pkg requestors phone	No
SYSLLPST	lastPromoSite	8	Last promotion site	No
SYSLLPLV	lastPromoLevel	2	Last promotion level	No
SYSLLPNM	lastPromoName	8	Last promotion name	No
SYSLLPDT	lastPromoDate	8	Last promotion date yyyymmdd	No

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)
SYSLLPID	lastPromoUserid	8	Last promotion userid	No
SYSLPAC	lastPromoAction	2	FP – Full Promotion FD – Full Demotion SP – Selective Promotion SD – Selective Demotion	No
Repeated Group (Variable Length)				
SYSLLTPN	pkgAllocLibType	3	Library type which is physically allocated to the components package	No
Repeated Group (Variable Length)				
SYSLFROM	pkgSyslib.whereFrom	1	S – staging P – Promotion B - Baseline	No
SYSLPKG or SYSLATTR_PKG_S	pkgSyslib.stagingPkg	10	If SYSLFROM=S then this is the package to which the staging library belongs	No
SYSLPRST or SYSLATTR_SITE_P	pkgSyslib.promoSite	8	If SYSLFROM=P then this is the promotion site to which this promotion library belongs	No
SYSLPRLV or SYSLATTR_LEVEL_P	pkgSyslib.promoLevel	2	If SYSLFROM=P then this is the promotion level (01-99) to which this promotion library belongs	No
SYSLPRNM or SYSLATTR_NAME_P	pkgSyslib.promoName	8	If SYSLFROM=P then this is the promotion name (nickname) to which this promotion library belongs	No
SYSLAPPL	pkgSyslib.appl	4	Application for this library	No
SYSLLIBT	pkgSyslib.libType	3	Library type for this library	No
SYSLSPEC	pkgSyslib.specialProcessInd	1	Special processing indicator for this libtype (e.g. D – Db2 and I – IMS)	No
SYSLSUBT	pkgSyslib.specialSubType	1	Special processing subtype	No
SYSLDSNM	pkgSyslib.name	Varchar	The syslib library or directory name	No
Returned by Exit				
SYSLGO	proceed	3	Set to 'NO' to stop the file tailoring process	Yes
SYSLLONG	longMsg	128	Long error message text	Yes

LE-Language Variable Name	REXX Variable Name	Length	Purpose	Modifiable (Yes/No)
SYSLCHNG	dataChanged	3	This field must be set to YES if you wish to return changed values to ZMF.	Yes
Repeated Group (Variable Length)				
SYSLIBNM	adjustedSyslib	Varchar	Changed syslib library or directory name	Yes

Component Standard Language Extraction Process (SYSL00XL, SYSL01XL)

There are two exits available for this process (which is not directly related to the package syslib process but the exits have been defined here to avoid a proliferation of functional areas). On input they take the following variables (REXX):

- function
- externalName
- debugCall
- callOrigin
- zmfSubs
- db2Subs
- userid
- component
- componentType
- application
- buildProc
- language

Note the following:

- buildProc will only be populated for SYSL01XL and only for like-SRC components.
- language will only be populated for SYSL00XL if CMNEX038 has already set it.
- The only variable that can be passed back to the service is language.

The equivalent COBOL fields are as follows:

```
STDLFUNC          PIC X(8).
STDLDBUG          PIC X(1).
STDLORGN          PIC X(3).
STDLZMFS          PIC X(1).
STDLPDB2          PIC X(4).
STDLUSER          PIC X(8).
STDLEXTN          PIC X(256).
STDLGO            PIC X(3).
STDLSHRT          PIC X(24).
```

```
STDLLONG          PIC X(128).
STDLCHNG          PIC X(3).
STDLCOMP          PIC X(256).
STDLLTYP          PIC X(3).
STDLAPPL          PIC X(4).
STDLPROC          PIC X(8).
STDLLANG          PIC X(8).
```

The equivalent PL/I fields are as follows:

```
STDLFUNC    CHAR(8),
STDLDBUG    CHAR(1),
STDLORGN    CHAR(3),
STDLZMFS    CHAR(1),
STDLPDB2    CHAR(4),
STDLUSER    CHAR(8),
STDLEXTN    CHAR(256),
STDLGO      CHAR(3),
STDLSHRT    CHAR(24),
STDLLONG    CHAR(128),
STDLCHNG    CHAR(3),
STDLCOMP    CHAR(256),
STDLLTYP    CHAR(3),
STDLAPPL    CHAR(4),
STDLPROC    CHAR(8),
STDLLANG    CHAR(8);
```

Sample exits are provided which show how to display the data items passed to SYSL00XL and SYSL01XL. They also show how to set a standard language based on certain selection criteria (appl and libtype in this case). The exits are:

- HXCSTDL - COBOL
- HXPSTDL - PL/I
- HXRSTDL - REXX

Scratch/Rename

This section describes the scratch/rename functional area of the high-level language exits. The 4-character exit name identifier is SCRN.

Select option U Scratch/Rename from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to define exits for the scratch rename function:

```
CMNHLLMM          HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All            Full list

2  Build          Component checkin, build, recompile, relink, delete
3  Package Create Initial create of a package
4  Package Update Subsequent update of package attributes
5  File Tailoring Define customized ISPF variables for file tailoring
6  Checkout       Component Checkout from baseline/promotion
7  Promote/Demote Promotion and demotion of components
8  Audit          Audit job submission and audit process
9  Freeze         Package freeze and selective unfreeze/refreeze
A  Approve/Reject Package approve and reject
R  Revert/Backout Package revert and backout
S  Package Syslib Package syslib list service
U  Scratch/Rename Utility functions

M  Miscellaneous  HLLX procedure name
Z  Modify         Issue Reload, Detach, or Attach modify commands
```

In response, the HLL Exit Definition (CMNHLLMN) panel is displayed. Here is how a sample panel might look:

```
CMNHLLMN                      HLL Exit Definition             Row 1 to 8 of 8
Command ===> ____________________________________________ Scroll ===> CSR

Internal External  + Active  1=LE    Description  +
  Name     Name              2=REXX  Debug Userids  +
-------------------------------------------------------------------------------
SCRN00XM _________     NO    2       pre  service call for pkg_util
 Debug:  _________     NO    2       ____________________________________
-------------------------------------------------------------------------------
SCRN01XM _________     NO    2       post service call for pkg_util
 Debug:  _________     NO    2       ____________________________________
-------------------------------------------------------------------------------
SCRN0101 _________     NO    2       post package selection
 Debug:  _________     NO    2       ____________________________________
-------------------------------------------------------------------------------
SCRN0002 _________     NO    2       pre  baseline selection
 Debug:  _________     NO    2       ____________________________________
-------------------------------------------------------------------------------
SCRN0102 _________     NO    2       post baseline selection
 Debug:  _________     NO    2       ____________________________________
-------------------------------------------------------------------------------
SCRN0103 _________     NO    2       post baseline member list
 Debug:  _________     NO    2       ____________________________________
-------------------------------------------------------------------------------
SCRN0104 _________     NO    2       post package member list
 Debug:  _________     NO    2       ____________________________________
-------------------------------------------------------------------------------
SCRN0105 _________     NO    2       post libtype selection list
 Debug:  _________     NO    2       ____________________________________
****************************** Bottom of data ********************************
```

The panels around which exit points will be placed are listed below. The internal exit name (also know as function code) is SYSL0*pnn*, where:

- *p*=0 is the pre-exit
- *p*=1 is the post-exit
- *nn* is an alphanumeric identifier relating to the panel for which the exit is taken.

The pre-exit is taken before the panel is displayed. The post-exit is taken after the panel has been displayed.

An internal exit name of SCRN0*p*02, for example, means that both pre- and post-exits exist. That is, the name of the pre-exit is SCRN0002 and the name of the post-exit is SCRN0102. If it makes no sense to have a pre-exit, the internal name is given as SCRN0101 (post-exit only). If it makes no sense to have a post-exit, the internal name is given as SCRN0001 (pre-exit only).

Most table displays have only post-exits. That is, we do not want to have a pre-exit that manipulates the lists that ZMF generates. We may want to have a post-exit to validate the selections that the user makes from the lists.

Sample exits are provided which show how to list all the information incoming to these exits. Note that not all information is available to all exit points. Those early in the dialog will not have as much information as back-end exit points.

The samples provided are:

CMNZMF.SAMPLES Library Member	**Description**
HXCSCRN	COBOL example
HXPSCRN	PL/I example
HXRSCRN	REXX example

There is a single data structure passed to all of these exits.

The data interface looks like this:

LE-Language variable name	REXX variable name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
SCRNFUNC	function	8	Internal exit name	No	
SCRNDBUG	debugCall	1	Debug exit call (Y/N)	No	
SCRNORGN	callOrigin	3	ISPF = SPF XML Service = XML ZDD = ZDD ZMF4ECL = ECL	No	
SCRNZMFS	zmfSubs	1	ZMF subsystem character	No	
SCRNPDB2	db2Subs	4	Default Db2 subsystem for this ZMF	No	

LE-Language variable name	REXX variable name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
SCRNUSER	userid	8	Userid for function calling this exit	No	
SCRNEXTN	externalName	256	External routine name defined for this exit	No	
SCRNPKGN	packageId	10	The package being acted on	No	
SCRNPSTA	packageStatus	3	Package status (DEV, FRZ, etc)	No	
SCRNPINS	packageInsDate	8	Pkg Install Date yyyymmdd	No	
SCRNWRQN	packageWorkRequest	12	Package Work Request	No	
SCRNDEPT	packageDepartment	4	Package Department	No	
SCRNOPTN	optionRequested	1	1=baseline, 2=package	Yes	001
SCRNRQST	requestType	1	s=scratch, r=rename, d=delete	Yes	013
SCRNCOMP	componentName	256	Component name	Yes	010
SCRNLTYP	componentType	3	Component library type	Yes	011
SCRNNWNM	componentNewName	256	Component New Name (rename)	Yes	012
SCRNVVMM	verModLevel	5	Version.mod level	No	
SCRNCRDT	createDate	8	Member create date	No	
SCRNCHDT	changeDate	8	Member change date	No	
SCRNCHGT	changeTime	6	Member change time	No	
SCRNCSZE	memberSize	5	Member change size	No	
SCRNUSRN	Username	8	User name	No	
SCRNLSZE	loadSize	6	Load member size	No	
SCRNLTTR	loadTtr	6	Load member TTR	No	
CKOTALAS	loadAlias	8	Load member alias	No	
SCRNAC	loadAuthCode	2	Authorisation code	No	
SCRNRM	loadRmode	3	Load Residency mode	No	
SCRNAM	loadAmode	3	Load Addressing mode	No	
SCRNATTR	loadAttributes	8	Load attributes	No	
SCRNGO	proceed	3	Set to 'NO' to stop the process	Yes	
SCRNLOKD	dataLocked	3	Fields locked? (YES/NO)	Yes	
SCRNSHRT	shortMsg	24	Short error message text	Yes	
SCRNLONG	longMsg	128	Long error message text	Yes	

LE-Language variable name	REXX variable name	Length	Purpose	Modifiable (Yes/No)	Cursor Field No.
SCRNCURS	cursorField	3	For ISPF where you wish the cursor to be placed on return to the panel display. The values relating to each field are shown in this table.	Yes	
SCRNCHNG	dataChanged	3	This field must be set to YES if you wish to return changed values to ZMF.	Yes	

Miscellaneous

Select option M Miscellaneous from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to specify or change the name of the HLLX procedure that is associated with the current ZMF instance:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> ____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

The HLL Exit Miscellaneous Parameters (CMNHLLMP) panel is displayed. There is currently only one entry on this panel: the name of the HLLX procedure that is associated with the current ZMF instance:

```
CMNHLLMP          HLL Exit Miscellaneous Parameters

Command ===>                                          Scroll ===>

HLLX procedure name . . SERDHLLI
```

Modify

This section describes the Modify feature of the high-level language exits. This now allows ChangeMan Administrators to modify the started task without resorting to using SDSF or operator console commands and typing in commands that may be prone to error or restricted.

Select option Z Modify from the HLL Exit Definition - Function Selection (CMNHLLMM) panel to Reload, Detach or Attach exit definitions:

```
CMNHLLMM           HLL Exit Definition - Function Selection
Option ===> _____________________________________________

1  All             Full list

2  Build           Component checkin, build, recompile, relink, delete
3  Package Create  Initial create of a package
4  Package Update  Subsequent update of package attributes
5  File Tailoring  Define customized ISPF variables for file tailoring
6  Checkout        Component Checkout from baseline/promotion
7  Promote/Demote  Promotion and demotion of components
8  Audit           Audit job submission and audit process
9  Freeze          Package freeze and selective unfreeze/refreeze
A  Approve/Reject  Package approve and reject
R  Revert/Backout  Package revert and backout
S  Package Syslib  Package syslib list service
U  Scratch/Rename  Utility functions

M  Miscellaneous   HLLX procedure name
Z  Modify          Issue Reload, Detach, or Attach modify commands
```

```
CMNHLLMC                  HLL Exit MODIFY commands
Option ===> _____________________________________________


1  Reload          Reload active HLL exit definitions
2  Detach          Stop HLLX
3  Attach          Start HLLX
```

This function allows you to request ZMF to issue MVS modify commands related to the operation of the HLL exit facility. For example, you can reload the active HLL exit definitions from those saved in the package master file. The valid options are:

1 - Reload active HLL exit definitions - refreshed all exits.

F stcname,CMN,RELOAD,HLLX

2 - Stop HLLX - stops the assocaiated HLLX address space.

F stcname,CMN,DETACH,HLLX

3 - Start HLLX - starts the assocaiated HLLX address space.

F stcname,CMN,ATTACH,HLLX

Each should result in the short message **Service completed** and a long message similar to **CMN8700I - HLLX Modify RELOAD service completed** (RELOAD will be DETACH or

ATTACH according to command selected) and the normal messages in the started task log for example:

```
SER0850I Operator command: CMN,RELOAD,HLLX
CMN8485I CMNSTART HLLX active exit table has been reloaded.
CMN8492I Start of HLLX active exits list:
CMN8494I IntName  Typ Env  External Name        Debug Ids
CMN8495I PCRE00PU STD REXX HXRSUBM3
CMN8493I End of HLLX active exits list.
```

The system log will show a message similar to:

```
MODIFY SERT8813.SERT8813,CMN,RELOAD,HLLX
```

Index

R

S

T

U

V