

# ChangeMan ZMF

## Db2 Option Getting Started Guide

8.3

# Table of Contents

---

About this Guide	5
Guide to ChangeMan ZMF Documentation	5
Choosing the Right Installation/Upgrade Manual	9
Online Help	10
Typographical Conventions	10
Introduction	12
Introduction	12
ChangeMan ZMF Db2 Option	12
Db2 Option Concepts	13
Db2 Option and Component Management	18
Configuring the Db2 Option	23
Configuring the Db2 Option	23
Overview	23
Apply a Db2 Option License	25
Bind Db2 Plan and Package and Grant access	26
Update Global Administration	27
Update Application Administration	32
Configure Db2 Option Global Administration	37
Define Application Logical Subsystems	62
Customize Skeletons for Db2	83
Installation in Other Db2 Subsystems	85
DB2 Component Processing	86
Db2 Component Processing	86
Library Types and Sub Types	86
CREATE versus ALTER	89
Component Processing Summary	90
Native SQL SP Lifecycle	104
Native SQL SP Lifecycle	104
Checkin/Stage	104
Promote	107

Demote	108
Install	109
Backout	110
Skeleton changes (overview)	111
Templating Examples	112
Templating Examples	112
Templated BIND Command Parameters	112
Templated DDL/SQL	113
Templating Examples	113
BIND PLAN Example	116
BIND PACKAGE Example	121
General token templates	126
CMNDB2PL - BIND Utility	140
CMNDB2PL - BIND Utility	140
Introduction	140
CMNDB2PL DD Statements	141
CMNDB2PL Operation	142
Keyword Control Statements	144
How CMNDB2PL Relates to ChangeMan ZMF	150
CMNDB2PL Return Codes and Messages	151
Sample CMNDB2PL Report	152
Secondary Binding	154
Stored Procedure Utilities	155
Stored Procedure Utilities	155
Introduction	155
CMNDB2AV	156
CMNDB2DQ	158
CMNDB2DD	159
CMNDB2SL	165
CMNDB2TR	167
CMNDB2DR	168

Stored Procedure Walkthrough	172
Bind Service Support	181
Bind Service Support	181
Installation and Configuration	181
Process Overview	189
Db2 Option User Exits	199
Db2 Option User Exits	199
CMNEX101 Bind Control Statement Processor	199
CMNEX103 Bind Control Statement Triage	203
CMNDB2DD - HLL exit	204
ISPF Tables and Variables	209
ISPF Tables and Variables	209
ISPF Tables and Variables	209
Single Entry Control Variables	213
Transaction Codes	215
Transaction Codes	215
Detailed Job List	215
Miscellaneous Transactions - at Either Site	220
Examples	222
Examples	222
Native SQL SP Versions and Bind Deploy	222
Support Use of zFS File Type for SP Components	238
Glossary	242
Legal Notice	243
Third-Party Notices	243
Specific notices	243

# 1. About this Guide

---

The ChangeMan ZMF Db2 Option Getting Started Guide provides instructions for installing and using the Db2 Option of ChangeMan ZMF to manage changes to application Db2 components.

This document is intended for ChangeMan ZMF installers, administrators, and Db2 data base administrators.

## Before You Begin

---

See the Readme for the latest updates and corrections for this manual.

## Navigating this Book

---

- Chapter 1 - Contains information about this manual.
- Chapters 2-3 describe the concepts behind the Db2 Option and how to install and configure it to meet your needs.
- Chapters 4-6 describe ChangeMan ZMF processing of Db2 components.
- Chapters 7-8 and Appendixes A-C provide information about the components in the Db2 Option so you can customize the option to fit your needs.
- Chapter 9 describes additional bind processing required for native Db2 REST services.
- Appendix D discusses examples with templates and Native SQL Stored Procedures(SP).

## Guide to ChangeMan ZMF Documentation

---

The following sections provide basic information about ChangeMan ZMF documentation.

## ChangeMan ZMF Documentation Suite

Manual	Description
Administrator's Guide	Describes ChangeMan ZMF features and functions with instructions for choosing options and configuring global and application administration parameters.
Customization Guide	Provides information about ChangeMan ZMF skeletons, exits, and utility programs that will help you to customize the base product to fit your needs.
Db2 Option Getting Started Guide	Describes how to install and use the Db2 Option of ChangeMan ZMF to manage changes to Db2 components.
ERO Concepts	Discusses the concepts of the 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 Exits Getting Started Guide	Explains how to configure and call the high-level language exits.
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 and 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.

<b>Manual</b>	<b>Description</b>
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.x and 8.x to version 8.2 Patch 6.
Online Forms Manager (OFM) Option Getting Started Guide	Explains how to install and use the OFM option of ChangeMan ZMF.
ChangeMan ZMF.	
REST Services Getting Started Guide	Getting Started Guide for ZMF REST Services.
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 Quick Reference	Provides a summary of the commands you use to perform the major functions in the ChangeMan ZMF package life cycle.

Manual	Description
ZMF Web Services User's 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/](http://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.

## Choosing the Right Installation/Upgrade Manual

Choose the manual that fits your situation when installing or upgrading ChangeMan ZMF.

Your task	Manual to use
Installing ChangeMan ZMF for the first time	ChangeMan ZMF Installation Guide
Building a new ChangeMan ZMF 8.2 Patch 6 instance from scratch	ChangeMan ZMF Installation Guide
Upgrading from ChangeMan ZMF 7.x or 8.x to version 8.2 Patch 6	ChangeMan ZMF Migration Guide (this manual)

### Important

Always see the most current Readme for your ChangeMan ZMF release in case it contains documentation updates for the installation/upgrade manual you use.

## ChangeMan ZMF Release Notes

High-level descriptions of the enhancements that are delivered in the ChangeMan ZMF 8.2 major version release and in all subsequent ZMF 8.2.x maintenance and patch releases are included in the "Features and Fixes" section of the latest ChangeMan ZMF 8.2 Patch 6 Readme.

## Online Help

---

Online help is the primary source of information about ChangeMan ZMF. Online help is available as a tutorial, through help panels, 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 panel 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 Panels

---

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

### Online Error Messages

---

If you make an invalid entry on a ChangeMan ZMF panel, or if you make an invalid request for a function, a short error message is displayed in the upper right corner of the panel.

Press PF1 to display a longer error message that provides details about the error condition.

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

## Typographical Conventions

---

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

Convention	Explanation
<i>italics</i>	Introduces new terms that you may not be familiar with and occasionally indicates emphasis.

Convention	Explanation
<b>bold</b>	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.
<i>monospace</i> <i>italics</i>	Indicates names that are placeholders for values you specify; for example, <i>filename</i> .
monospace bold	Indicates the results of an executed command.
vertical rule	Separates menus and their associated commands. For example, select File

#### Note

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

## 2. Introduction

---

This chapter introduces you to the Db2 Option of ChangeMan ZMF.

- [ChangeMan ZMF Db2 Option](#)
- [Db2 Option Concepts](#)
- [Db2 Option and Component Management](#)

### ChangeMan ZMF Db2 Option

---

The Db2 Option of ChangeMan ZMF consists of proprietary programs, ISPF skeletons, and additional ChangeMan ZMF administration parameters that let you:

- Manage Db2 components and objects with automated Software Change Management (SCM) processes in the change package lifecycle employed by ChangeMan ZMF.
- Automatically perform Db2 binds in local and remote Db2 test subsystems when you promote and demote application components.
- Automatically perform binds in production environments when you install or back out packages that contain Db2 components.
- Automate the processes required to implement stored procedures during promote, demote, Install, and back out.
- Automate the processes required to implement user defined functions and triggers during promote, demote, install, and back out.

#### Note

Db2 is not compatible with reusable ASIDs in z/OS. You should not use the z/OS START command parameter REUSASID=YES to start a ZMF instance that includes the Db2 Option.

## Db2 Option Not Compatible with Reusable ASIDs

---

The Db2 Option of ChangeMan ZMF calls Db2, which is not compatible with reusable ASIDs, so the Db2 Option is not compatible with reusable ASIDs. If you use z/OS START command parameter REUSASID=YES to start a ZMF instance where the Db2 Option is licensed, the Db2 Option will not work as expected.

To use the Db2 Option, restart the ZMF instance without the REUSASID=YES parameter in the START command. See the Knowledgebase [Solution S141854](#) for more information.

## Db2 Option Concepts

---

This section defines terms and concepts used by the ChangeMan ZMF Db2 Option.

### Core Db2 Option

---

The original support provided by the ZMF Db2 option addresses the automation of plan and package binds throughout the lifecycle. In more recent years, support for additional objects and processes has been introduced. This additional support requires the creation of a number of infrastructure items (Db2 option tables etc.) and the binding of a wider range of ZMF supplied packages. If you do not want to use this additional support, you do not need to put in place the extra infrastructure items that it requires. You can choose to keep your overhead low and concentrate on the original 'core' Db2 option support. In this section, the term 'Core ZMF Db2 option support' or, simply, 'core support' refers to the automation of program package and plan binds (through the use of program CMNDB2PL and supporting admin).

### Physical and Logical SubSystem

---

In the ChangeMan ZMF Db2 Option, the Db2 subsystems where test and production Db2 components run are identified as physical subsystems. Each Db2 Option physical subsystem is associated with a local or remote site.

If you have enough Db2 resources, you can have a separate Db2 subsystem for production and for every test environment. Every program can be bound under the same plan name and package collection ID in each Db2 subsystem without conflict.

Most ChangeMan ZMF user sites, however, use the same Db2 subsystem for several test environments. Some user sites use a single Db2 subsystem for production and test. Where the same program is run in multiple environments in the same Db2 subsystem, it must be bound under different package collection IDs and plan names to avoid conflict.

Qualifier and bind owner may also be different for the different uses of the program. The same stored procedures can be registered in the Db2 catalog under different schemas.

The ChangeMan ZMF Db2 Option partitions a physical Db2 subsystem with logical subsystems. A logical subsystem is a set of rules for changing plan names, package location, package collection ID, qualifier, bind owner, schema, and WLM environments to provide unique entries in the Db2 catalog. A logical subsystem also includes rules for managing stored procedures and triggers, as well as what to do in promotion when a bind fails.

Logical subsystems are assigned a name, which is sometimes called a nickname. Each logical subsystem is associated with a single physical subsystem.

Recent enhancements address the requirements of a modern Native SQL development lifecycle such as when initiated via IBM® Data Studio (although IBM Data Studio is not a pre-requisite, the same Native SQL code can be hand written directly via an ISPF edit session).

## Active Libraries

Automated Db2 Option functions are activated when libraries managed by ChangeMan ZMF are changed in promotion and production environments. These libraries are defined as active libraries in the Db2 Option. Each active library is associated with a logical subsystem. When the contents of an active library are changed, Db2 Option functions are invoked and executed according to the rules defined in the logical subsystem.

### Bind Active Libraries

When a Db2 program is changed, or when the BIND command that references a DBRM is changed, the DBRM for that program must be bound in the Db2 subsystem where the program is executed. DBRM libraries and libraries containing BIND command members are defined as BIND active libraries to trigger Db2 binds at promote, demote, install, and backout.

### SQL Active Libraries

When a stored procedure, user-defined function, or trigger is added or changed, SQL must be processed to create or change the definition in the Db2 catalog. Stored procedures may be stopped and started to activate changes. When a database trigger is changed, it may be necessary to recreate other triggers to maintain the original firing order. This special processing is invoked by defining certain libraries as SQL active libraries. Promotion and production libraries that contain stored procedures, triggers, and user-defined functions are defined as SQL active libraries.

### Bind Service Active Libraries

Copying into a Bind service active library causes ChangeMan to drive the process required to define a Db2 RESTful service at the appropriate Db2 subsystem(s).

## Db2 Library Subtypes

---

Db2 library subtypes invoke special processing for Db2 components. There is a discussion with more information in the [Define Global Db2 Library Subtypes](#) section under **Configuring the Db2 Option**.

## Templates

---

In the ChangeMan ZMF Db2 Option, you create transformation rules to express the difference between BIND commands (etc.) executed in various environments such as production and test. When you promote, demote, install, or back out a Db2 program, the Db2 Option applies these transformation rules to model BIND commands in staging or baseline libraries to create BIND commands suitable for the target environment. The binds (etc.) are then performed automatically. Bind service definitions are also templated in a similar way, though using a different process to plans and packages.

Transformation rules can also be defined for parameters in DDL for stored procedures, triggers, and user-defined functions. When you promote, demote, install, or back out a Db2 stored procedure, trigger, or user-defined function, the Db2 Option applies the transformation rules to model DDL statements in staging or baseline libraries to generate SQL suitable for the target environment. The modified DDL are then actioned automatically.

There are two types of template definitions, named field templates and general token templates. The former are applied to a set of commonly templated bind/DDDL parameters and are described in this section. The latter are more general and allow you to search for a parameter keyword before applying a template to the values on that general keyword. General token templating is described more fully in the [Define Application Logical Subsystems](#) section.

Transformation rules are defined in templates in the ChangeMan ZMF Db2 Option. You define a set of templates for each Db2 Option logical subsystem.

The model BIND commands and DDL to which these templates are applied are often the BIND commands and SQL you use in one of your production environments. If you use these production components for your models, the templates are empty for the logical subsystem corresponding to the production environment.

You can modify the following parameters in a BIND command by using templating:

- PLAN Name
- PACKAGE Location
- PACKAGE Name/Collection ID
- Owner
- Qualifier

If the owner or qualifier parameters are missing from the BIND command, you can insert these parameters by coding Insert values in the logical subsystem definition. Insert values are applied during templating when owner and qualifier are missing from the BIND command and the following CMNDB2PL control cards are present: AUTHORITY=OWNER,INSERT and INSERTQUAL.

You can modify the following parameters in the DDL for stored procedures, triggers, and user defined functions by using templating:

- Schema
- Collection ID
- Qualifier
- WLM environment
- Owner

The following parameters can be templated directly in bind service definitions:

- Collection ID
- Qualifier
- Owner

The template algorithms are Insert, Deploy, Search and Replace, and Positional Character Replacement.

## Insert

Insert applies only to BIND templates owner and qualifier. If these keywords are missing from a BIND command, and control statement parameters AUTHORITY=OWNER,INSERT *Db2 Option Concepts*

and INSERTQUAL are input to the Plan Lookup program CMNDB2PL, the values specified in the logical subsystem template are added to the BIND command.

## Deploy

Deploy applies only to SQL templates LOCATION, owner and qualifier. They are used if the

BIND DEPLOY mechanism is chosen for Native SQL stored procedures. The 'DEPLOY Location' is used to route the execution of the bind deploy command from the target Db2 subsystem back to the source Db2 subsystem for the deployment. OWNER and QUALIFIER may be specified on the bind deploy command itself and the values for these will be templated as normal. However, if the template process does not result in a nonblank value then anything specified in the DEPLOY templates for these parameters will be used instead.



## Search and Replace

BIND command parameters and DDL parameters are searched for a value specified in the logical subsystem template. If the string is located, it is replaced by another value specified in the template.

## Positional Character Replacement

The character in a particular position of a BIND command parameter or DDL parameter is replaced with a character specified in the logical subsystem template. Specified characters can be also be added at the end of BIND command parameters or DDL parameter.

## Templates And Change Management

---

Validated templates are essential to software change management for BIND commands or DDL because you cannot test these components in a production environment. Validated templates provide an automated transformation for these components that ensures that if they work for promotion (test), they will also be valid for production.

### Plan/package Lookup

Plan/package Lookup program CMNDB2PL is included in batch jobs that promote, demote, install, or back out change packages that contain Db2 programs. The Plan/package Lookup program performs two functions:

- Finds the Db2 plans and packages that contain the DBRM staged in the change package and locates the PDS members that contain BIND commands to bind the DBRM.
- Applies templates to the BIND commands to transform them for use in the target Db2 environment.

The Plan/package Lookup program searches the Db2 SYSPACKAGE and SYSDBRM tables to find packages and plans where the DBRM in the change package was bound previously. The program then looks in staging libraries for the members that contain BIND commands for the list of plans and packages. If the BIND command members are not in staging libraries, then the Plan Lookup program looks in promotion and baseline libraries, in that order.

See the [CMNDB2PL - BIND Utility](#) section for a more detailed description of how the Plan/package Lookup program works and the control statements that can alter its behavior.

### BIND Fail

When you install a Db2 component into your production environment, you want the install process to fail if the Db2 binds fail.

However, the same may not be true for your test environments. You can set a parameter in each logical subsystem to allow the promotion process to complete successfully even if the Db2 bind jobs fail.

# Db2 Option and Component Management

---

The Db2 Option of ChangeMan ZMF manages Db2 components through a rules-based life cycle for development, test, and install that ensures component and application integrity. The Db2 Option automates two classes of functions for Db2 components:

- Db2 binds
- Db2 object management for stored procedures, triggers, and user defined functions.

## Important

ChangeMan ZMF programs in the Db2 Option assume that BIND commands and DDL for stored procedures, triggers, and user-defined functions are syntactically correct. BIND commands that are input to program CMNDB2PL are parsed with IBM service routine IKJPARS to ensure that CMNDB2PL processing is synchronized with IBM changes to BIND keyword operands.

## Bind Processing

---

The ChangeMan ZMF Db2 Option binds the DBRM for programs in change packages when the component is promoted or demoted, and when the package is installed or backed out.

Features of automated bind processing:

- The Db2 catalog is searched for plans and packages that reference staged DBRM, and those plans and packages are bound.
- Parameters in BIND PLAN and BIND PACKAGE commands in staging libraries or baseline libraries can be modified according to fixed rules to adapt the BIND commands to the Db2 subsystems used for test and production.

## Stored Procedure Processing

---

Stored procedures are user-written application programs that can be called by SQL programs that run either locally or remotely on any platform supported by the IBM® Db2® UDB network. The Db2 Option of ChangeMan ZMF supports external stored procedures, external SQL stored procedures, and native SQL stored procedures.

### External Stored Procedures

These are programs coded in a traditional host language like assembler, COBOL, PL/I, C or C++, or REXX.

*Db2 Option and Component Management*

These component types are involved in managing external stored procedures:

- Stored procedure source
- Stored procedure Load
- DBRM
- Link edit control statements
- DSN BIND command
- Non-SQL stored procedure DDL (CREATE PROCEDURE)

External stored procedures are staged as like-source components. The source is processed by the Db2 precompiler to create a DBRM, then compiled, prelinked (for some languages), and link edited to create an executable load.

For a new external stored procedure, a link control member is staged to include required Db2 subroutines in the stored procedure load module. A BIND command member is staged to bind the DBRM at promotion and install. A CREATE PROCEDURE DDL is staged to define the stored procedure in the Db2 subsystem at promotion and install.

At promotion and install, the BIND command and CREATE PROCEDURE DDL are templated to adapt them to the target Db2 subsystem. The DBRM for the stored procedure is bound in the target Db2 subsystem. A DROP PROCEDURE is automatically issued, then the CREATE PROCEDURE DDL is executed to register the stored procedure in the Db2 catalog.

The stored procedure load module is copied to the target execution library. The VARY WLM,APPLENV=envname,REFRESH command is automatically issued to refresh the stored procedure executable in the WLM-managed address space.

The external stored procedure source, the BIND command, and the CREATE PROCEDURE DDL can be staged separately to make changes to the external stored procedure after the initial installation.

## External SQL Stored Procedures

External SQL stored procedures combine procedural code written in SQL with the CREATE PROCEDURE DDL that define the procedure in the Db2 subsystem. External SQL stored procedures are either hand coded or are generated by programs like the Db2 Data Studio which executes on a client platform such as Windows® and forms part of the Db2 Connect.

These component types are involved in managing SQL stored procedures:

- Stored procedure source
- Stored procedure Load
- DBRM
- Link edit control statements
- DSN BIND command

An SQL language stored procedure is staged as a like-source component. The entire component, including the CREATE PROCEDURE DDL and SQL procedural code, is translated by the Db2 precompiler into C code. The C code is then processed like a traditional external stored procedure module through the Db2 precompiler to create a DBRM, then compiled, prelinked, and link edited to create an executable load.

For a new SQL stored procedure, a BIND command member is staged to bind the DBRM at promotion and install.

At promotion and install, the BIND command is templated to adapt it to the target Db2 subsystem. The DBRM for the stored procedure is bound in the target Db2 subsystem. The SQL language stored procedure source is processed to extract the CREATE

PROCEDURE DDL. A DROP PROCEDURE is automatically issued, then the CREATE

PROCEDURE DDL is executed to register the stored procedure in the Db2 catalog. The VARY WLM,APPLENV=envname,REFRESH command is automatically issued to refresh the stored procedure executable in the WLM-managed address space.

The SQL language stored procedure source and the BIND command can be staged separately to make changes to the stored procedure after the initial installation.

## Native SQL Stored Procedures

A Native SQL stored procedure is one in which the DDL, the procedural logic, and the SQL statements are contained in a single component. Db2 builds and schedules the executable internally and no other input is required to define this object. A Native SQL stored procedure is staged as a like-PDS component. There is no transformation at stage, and no other component types are required. At promotion and install, the DDL is templated to adapt it to the target Db2 subsystem. Propagation to the target Db2 subsystem is via one of DROP/CREATE, ALTER ADD VERSION, or BIND PACKAGE DEPLOY mechanisms. Facilities are in place within the ZMF Db2 option to automate all of these deployment mechanisms.

## User-Defined Functions

A user-defined function (UDF) is defined to Db2 with the CREATE FUNCTION statement and can be referenced thereafter in SQL statements. User-defined functions can be used in place of or in addition to built-in functions. There are two major categories of UDFs: sourced and external.

### Sourced User-Defined Functions

Sourced user-defined functions are composed of existing built-in functions and previously defined user-defined functions. The definition of a sourced UDF is made entirely within a CREATE FUNCTION statement.

A sourced user-defined function is staged as a like-PDS component. No other components are required.

At promotion and install, the CREATE FUNCTION statement is templated to adapt it to the target Db2 subsystem. A DROP FUNCTION is automatically issued, then the CREATE FUNCTION statement is executed to define the sourced UDF in the Db2 subsystem.

## External User-Defined Functions

External user-defined functions are implemented by means of an externally written program and are managed in the Db2 Option like an external stored procedure.

These component types are involved in managing external user defined functions:

- Stored procedure source

- Stored procedure Load

*Db2 Option and Component Management*

- DBRM

- Link edit control statements

- DSN BIND command

- Non-SQL stored procedure definition (CREATE PROCEDURE)

External UDFs are staged as like-source components. If the source contains imbedded SQL, the source is processed by the Db2 precompiler to create a DBRM. The source is compiled, prelinked (for some languages), and link edited to create an executable load.

For a new external UDF, a link control member is staged to include required Db2 subroutines in the stored procedure load module. If the source contains imbedded SQL, A BIND command member is staged to bind the DBRM at promotion and install. A CREATE FUNCTION statement is staged to define the UDF in the Db2 subsystem at promotion and install.

At promotion and install, the BIND command and CREATE FUNCTION statement are templated to adapt them to the target Db2 subsystem. The DBRM for the UDF is bound in the target Db2 subsystem. A DROP FUNCTION is automatically issued, then the CREATE FUNCTION statement is executed to register the UDF in the Db2 catalog. The UDF load module is copied to the target execution library, and a VARY WLM,APPLENV=envname, REFRESH command is automatically issued to refresh the UDF executable in the WLMmanaged address space.

The external user defined function source, the BIND command, and the CREATE

FUNCTION statement can be staged separately to make changes to the external UDF after the initial installation.

## Database Triggers

---

A trigger is a set of SQL statements that is stored in a Db2 database and executed when a certain event occurs in a Db2 table.

A trigger definition is staged as like-PDS. Trigger definitions are not transformed at stage, and no other component types are required.

At promotion and install, the CREATE TRIGGER statement is templated to adapt it to the target Db2 subsystem. A DROP TRIGGER is automatically issued. Then the CREATE TRIGGER statement is executed to define the trigger.

The Db2 Option looks in the Db2 catalog to see if there are multiple triggers for the same table/event/time combination. Multiple triggers can be recreated in an order defined by the contents of the COMMENT ON field in the CREATE TRIGGER SQL to maintain the desired trigger firing order.

Triggers are defined in CREATE TRIGGER SQL statements only. There is no external equivalent.

## Db2 Object Dependency Report

---

The Db2 Object Dependency report is a batch report that analyzes stored procedures and user-defined functions for dependencies that will interfere with the automatic DROP that is issued before a CREATE is executed at promote, demote, install, or backout.

# 3. Configuring the Db2 Option

---

This chapter tells you how to set up ChangeMan ZMF and the Db2 Option to manage Db2 objects and application components that use Db2.

- [Overview](#)
- [Apply a Db2 Option License](#)
- [Bind Db2 Plan and Package and Grant access](#)
- [Update Global Administration](#)
- [Update Application Administration](#)
- [Configure Db2 Option Application Administration](#)
- [Customize Skeletons for Db2](#)
- [Installation in Other Db2 Subsystems](#)

## Overview

---

If you are installing ChangeMan ZMF for the first time, you can defer configuring the Db2 Option until later, after your DBA and application developers have agreed on how they want to manage Db2 components with ChangeMan ZMF. The configuration described in this chapter does not play any part in the processing of non-Db2 components through the ChangeMan ZMF package life cycle.

The following table appears throughout this book to tell you how to define Db2 objects and application components that use Db2. The table also shows you how the definitions relate to other definitions.

<b>Db2 Component</b>	<b>Like</b>	<b>Target Type</b>	<b>Sel Opt</b>	<b>Sub Typ</b>	<b>BIND/SQL/SERVICE</b>
Db2 Application Program Source	S	Db2 Program Load			
Db2 Application Program Load	L				B
DBRM	P		D	R	B

<b>Db2 Component</b>	<b>Like</b>	<b>Target Type</b>	<b>Sel Opt</b>	<b>Sub Typ</b>	<b>BIND/SQL/SERVICE</b>
BIND PLAN Command	P		D	B	B
BIND PACKAGE Command	P		D	P	B
External Stored Procedure Source	S	Stored Procedure Load			
External SQL Stored Procedure Source	S	Stored Procedure Load	D	Q	S
External Stored Procedure Load	L		D	S	B & S
Native SQL Stored Procedure	P		D	N	S
General DDL (e.g. CREATE PROCEDURE for external SP)	P		D	D	S
User Defined Function Definition	P		D	D	S
Trigger Definition	P		D	T	S
BIND Service command	P		D	V	V
Service GRANT command	P		D	V	V

\* Db2 Active Library specification for BIND plan/pkg (B), Process SQL (S), and Bind Service (V).



## Note

- Load libraries containing external stored procedures must be PDSE. This includes staging, promotion, baseline, and production libraries. If stored procedure load libraries are defined as PDS, a link edit for stage, recompile, or relink of the stored procedure may fail with message IEW2606S.
- The execution load library containing stored procedures (baseline or production) must be concatenated in the STEPLIB of the appropriate Workload Manager (WLM) managed address space.
- Bind service commands and the related grant components can be either PDSE or zFS based.

## Apply a Db2 Option License

If you license the Db2 Option at the same time that you license ChangeMan ZMF, the license for the option is applied when you apply the license for the base product. You do not have to take further action to enable the Db2 Option.

If you license the Db2 Option after you apply licenses for ChangeMan ZMF and other selectable options, use the SER10TY™ License Manager to add a license for the option. See the *SER10TY User Guide* for instructions on how to apply a license. The load modules, JCL, and other components that you need to run SER10TY are included in the SERCOMC libraries in the ChangeMan ZMF installer

After you have applied a license, shut down the SERNET started task where ChangeMan ZMF runs and restart the task.

**Then, follow these steps to verify that the Db2 Option is activated.**

1. Connect to ChangeMan ZMF through ISPF.
2. From the **Primary Option Menu** type =A.G.O on the Option line to jump to the **Global Selectable Options** panel CMNGBSOP:

## Important

Do not change the delivered names except to code the embedded subsystem. Program CMNDB2SQ issues an internal SET CURRENT PACKAGESET CMNx command that determines which DBRM is used for the ChangeMan ZMF instance that is issuing the command.

CMNGBSOP	GLOBAL Selectable Options
Option ==>	
2 Db2	Maintain Db2 information
3 INFO	Specify Info/Management change rule
4 OFM	Configure Online Forms Manager
5 IMS	Control Region IDs and Library Sub-Type information

If option **2 Db2** is highlighted, the activation is successful.

## Bind Db2 Plan and Package and Grant access

If you only intend to use the core functionality of the plan/package bind automation, you need only customize and run the DB2CORE sample JCL member (on each lpar and foreach Db2 subsystem that could be the target of a promotion or install action). You can then skip straight to the next section [Update Global Administration](#). Otherwise, follow the instructions below.

Member DB2OPTN in the CNTL dataset should be copied to your CUSTOM library and edited to define the tables required in Db2, to bind the Packages and the Plan used by the Db2 option, and grants the minimum permissions required by the Db2 option.

If you are going to use remote Db2, then the CMNDB2VB package must be bound with the relevant qualifier at both the local and all potential remote DB2s. In addition, at the local DB2, the CMNPLAN plan must be bound with a package list including all the locations of all the remote DB2s where the CMNDB2VB package may be used.

Refer to the comments in the JCL for details.

### Important

Do not change the delivered names except to code the embedded subsystem. Program CMNDB2SQ issues an internal SET CURRENT PACKAGESET CMNx command that determines which DBRM is used for the ChangeMan ZMF instance that is issuing the command.

### Important

Do not change the name of the plan.

For each remote site, copy the DB2OPTNR member and follow its instructions to bind the CMNDBSQ package into the CMNPLAN for use by the automatic bind processing (CMNDB2PL) at each remote site (promotion and install). Note that there is an extra section not run at the end - the last two steps, BINDPKG and BINDPLAN are not reached as there is a // null line after the GRANT step (look for "end of Job" comment). These two steps are there as examples should you wish to use BIND DEPLOY from two source subsystems - see the comments in the JCL.

## Note

If you are upgrading from 8.1 or earlier, you will need to copy the member

DB2OPTNC and customize per the notes within, in order to perform the conversion of the SQL package master data into the tables in Db2. This job must be run after the DB2OPTN jobs has been run and has defined the tables.

## Note

- The subsystem ID embedded in the Db2 package name makes the name unique for each ChangeMan ZMF instance in a Db2 subsystem.
- All ChangeMan ZMF Db2 programs are precompiled with VERSION(AUTO) so you can have multiple versions of the package in the Db2 catalog.
- Program CMNDB2SQ executes SQL that accesses the Db2 catalog. You can optimize the SQL that queries the SYSDBRM and SYSPACKAGE catalog tables by creating an index on both tables on the NAME column. If you choose to create the indexes, be sure to rebind the CMNDB2SQ package following the index creation.
- The collection id for the CMNDB2AT package must always be CMNZMF.
- The sample JCL members create tablespaces as UTS-Partition By Growth (UTS-PBG) with a MAXPARTITIONS value of 1. The default values for DSSIZE are sufficient for the tablespaces hosting the ZMF Db2 option admin tables. However, the final decision on the values you use should rest with your database administrators and be in compliance with your site standards.

## Update Global Administration

---

Add special library types, and add a language and procedure for SQL stored procedures, to global administration for the base ChangeMan ZMF product.

## Add Global Library Types for Db2

You assign Db2 Option functions to a library type with the Selectable Option field in the application library type definition and with the Sub-type field in Db2 Option library type definition.

This table shows you the kinds of components managed by the Db2 Option and the library type parameters that are required for each. Parameters that appear on library type definition panels are shown in **bold**.

<b>Db2 Component</b>	<b>Like</b>	<b>Target Type</b>	<b>Sel Opt</b>	<b>Sub Typ</b>	<b>BIND/SQL/SERVICE</b>
Db2 Application Program Source	**S	**Db2 Program Load			
Db2 Application Program Load	**L				B
DBRM	**P		**D	R	B
BIND PLAN Command	**P		**D	B	B
BIND PACKAGE Command	**P		**D	P	B
External Stored Procedure Source	**S	**Stored Procedure Load			
External SQL Stored Procedure Source	**S	**Stored Procedure Load	**D	Q	S
External Stored Procedure Load	**L		**D	S	B & S
Native SQL Stored Procedure	**P		**D	N	S
General DDL (e.g. CREATE PROCEDURE for external SP)	**P		**D	D	S
User Defined Function Definition	**P		**D	D	S

Db2 Component	Like	Target Type	Sel Opt	Sub Typ	BIND/SQL/SERVICE
Trigger Definition	**P		**D	T	S
BIND Service command	P		D	V	V
Service GRANT command	P		D	V	V

\* Db2 Active Library specification for BIND plan/pkg (B), Process SQL (S), and Bind Service (V).

Follow these steps to create global library type definitions for components managed by the Db2 Option:

1. From anywhere in the ChangeMan ZMF ISPF client, type **=A.G.2** on the **Command** or **Option** line and press **Enter**. The **Global Library Types Part 1 of 2** panel CMNCGLT0 is displayed.
2. Follow the instructions in the *ChangeMan ZMF Administrator Guide* to insert a library type row and create a new library type for each kind of Db2 component you will manage.

CMNCGLT0 Global Library Types Part 1 of 2 Row 5 to 41 of 41  
 Command ==> \_\_\_\_\_ Scroll ==> CSR

Lib	Order	Lke	Seq	Defer	Target	Sel
type Description	+				type	Opt
DBB Db2 BIND PLAN Commands	0	P	001	Y		D
DBR Db2 DBRM	0	P	001	Y		D
DOC Documentation	0	P		Y		
...						
OBJ Object module library	0	O		Y		
PKG Db2 Bind Package Command	0	P		Y		D
PRC Cataloged Procedures	0	P		Y		D
SDB Db2 Program Source	0	S	003	Y	LDB	D
SPD Db2 Stored Proc Definitions - Non-SQL	0	P		Y		D
SPN Db2 Stored Proc Source - Native SQL	0	P		Y		D
SPQ Db2 Stored Proc Source - SQL Language	0	P		Y	STL	D
SRC Source for Programs to be Linked Exec	0	S		Y	LOD	
SRS Source for subprograms to be Linked N	0	S		Y	LOS	
STL Db2 Stord Prod Load Modules	0	L		Y		D
STP Db2 Stored Proc Source - External Lan	0	S		Y	STL	D
TRG Db2 Trigger Definitions	0	P		Y		D
TST Test Library type	0	P		Y		
UDF Db2 User-Defined Function Definitions	0	P		Y		D
...						
ZSS Shared Baseline Subprogram Source	0	S		Y	ZLS	

 **Important**


On library types for stored procedure load modules, set the **Data Set Type** field to **LIBRARY** (PDSE) on the **Global Library Types Part 2 of 2** panel. If stored procedure load libraries are defined as PDS, a link edit for stage, recompile, or relink of the stored procedure may fail with message IEW2606S.

Type **S** on the Line Command for each new library type row to display the **Global Library Types Part 2 of 2** panel. Note that to fully support native SQL SPs generated by Data Studio, you need to use VB and LRECL 255 records.

3. Exit the **Global Library Types** panels and save your changes.

## Add Global Language and Procedure for External SQL SPs

This is an optional step, only required if you need to continue to support External SQL SPs. External SQL stored procedures are processed through the Db2 precompiler to translate the source into the C language, which is then processed like an external stored procedure. Add a language for SQL stored procedures and add stage procedure CMNSQL.

 **Note**

Do not confuse the CMNSQL procedure with the CMNCEE procedure that you use for external stored procedures written in C. The CMNSQL procedure imbeds skeleton CMN\$\$SQP to translate SQL code to C source before compiling and link editing the C source.

1. From anywhere in the ChangeMan ZMF ISPF client, type **=A.G.3** on the **Command** or **Option** line and press **Enter**. The **Global Language Names** panel CMNGGLNG is displayed.
2. Insert a row and type a **Language** for SQL stored procedures. Language **SQL** is used in this example.

```
CMNGGLNG                      Global Language Names                      Row 1 to 9 of 9
Command ==>> _____ Scroll ==>> CSR

  Language Order
  ASM      0
  C        0
  COBOL    0
  COBOLE   0
  COBOL2   0
  JAVA     0
  PLI      0
  PLIE     0
  SQL      0
***** Bottom of data *****
```

3. Exit the **Global Language Names** panel and save your changes.
4. From anywhere in the ChangeMan ZMF ISPF client, type **=A.G.4** in the **Command** or **Option** line and press **Enter**. The **Compile Procedure List** panel is displayed.
5. Insert a row and type **\*** in the **Language** field to display the **Language Selection List** panel. Select the new language for SQL stored procedures from the **Language Selection List** panel.

```
CMNPRCNN                      Compile Procedure List                      Row 1 to 21 of 21
Command ==>> _____ Scroll ==>> CSR

  Language Procedure Description Order
  ASM      CMNASM  Stage assembler source           0
  ASM      CMNASMOB Stage assembler source to object    0
  ASM      CMNASM2L Stage assembler source w/ 2 link edit 0
  ...
  PLI      CMNPLI   Stage PL/I source                   0
  PLI      CMNPLIOB Stage PL/I source to object          0
  PLIE     CMNPLIE  Stage Enterprise PL/I source        0
  SQL      CMNSQL   Translate, compile, and link SQL Stored Proc 0
***** Bottom of data *****
```

6. Type **CMNSQL** in the **Procedure** field and type a description in the **Description** field.
7. Exit the **Compile Procedure List** panel and save your changes.

## Update Application Administration

---

Add library types, a language and procedure for SQL stored procedures, baseline libraries, and production libraries to all applications where you want to manage Db2 components.

In any particular application, you only need application administration entries for the kinds of components you want to manage in that application.

### Important

Libraries containing stored procedure load modules must be PDSE. This includes staging, promotion, baseline, and production libraries. If stored procedure load libraries are defined as PDS, a link edit for stage, recompile, or relink of the stored procedure may fail with message IEW2606S.

## Add Application Library Types for Db2

---

Follow these steps to add global library type definitions in each application with components managed by the Db2 Option:

1. From anywhere in the ChangeMan ZMF ISPF client, type **=A.A.2** on the Command or Option line and press **Enter** to display the **application - Library Types Part 1 of 2** panel.
2. Follow the instructions in the *ChangeMan ZMF Administrator Guide* to copy global library type definitions into the **application - Library Types Part 1 of 2** panel. See the Db2 library types in this example:



```

CMNCLLT0          ACTP - Library Types Part 1 of 2          Row 1 to 28 of 28
Command ==>> _____ Scroll ==>> CSR

  Lib              Order Lke Seq Defer Target Sel
  type Description +          type  Opt
CPY  Copybooks          0    C 001  Y
CP2  Copybooks for Utilities 0    C 002  Y
CTC  Control Statements    0    P    Y
DBB  Db2 BIND PLAN Commands 0    P 001  Y      D
DBR  Db2 DBRM             0    P 001  Y      D
...
JVS  HFS - JAVA source type  0    S    Y  JVL
JVT  HFS - text type         0    p    Y
LCT  Linkedit Control Cards  0    P    Y
PKG  Db2 Bind Package Commands 0    P    Y      D
PRC  Cataloged Procedures    0    P    Y      D
SDB  Db2 Program Source      0    S 003  Y  LDB
SPD  Db2 Stored Proc Definitions - Non-SQL 0    P    Y      D
SPN  Db2 Stored Proc Source - Native SQL  0    P    Y      D
SPQ  Db2 Stored Proc Source - SQL Language 0    p    Y  STL  D
SRC  Source for Programs to be Linked Exec 0    S    Y  LOD
SRS  Source for subprograms to be Linked N 0    S    Y  LOS
STL  Db2 Stored Proc Load Modules  0    L    Y      D
STP  Db2 Stored Prod Source - External Lan 0    S    Y  STL  D
TRG  Db2 Trigger Definitions    0    P    Y      D
TST  Test Library type         0    p    Y
UDF  Db2 User-Defined Function Definitions 0    P    Y      D
...
***** Bottom of data *****

```

3. Type **S** on the line command for the new application library types to display the **application - Library Types Part 2 of 2** panel. Note that to fully support native SQL SPs generated by Data Studio, you need to use VB and LRECL 255 records.

- a. Adjust the parameters from the global definition to fit the application, if necessary.
- b. For libraries containing stored procedure load modules, the **Data Set Type** field should be **LIBRARY** (PDSE).

4. Exit the **application - Library Types** panels and save your changes.

## Add Application Language and Procedure for External SQL SPs

This is an optional step, only required if you need to continue to support External SQL SPs.

Follow these steps to add a language and procedure for External SQL stored procedures to every application that will manage those components.

1. From anywhere in the ChangeMan ZMF ISPF client, type **=A.A.3** on the **Command** or **Option** line and press **Enter**. The **application - Language Names** panel is displayed.
2. Follow the instructions in the *ChangeMan ZMF Administrator Guide* to copy the global language name for SQL started procedures into the **application - Language Names** panel CMNCLLNG.

Language **SQL** is used in this example.

```

CMNCLLNG          ACTP - Language Names          Row 1 to 8 of 8
Command ==> _____ Scroll ==> CSR

Language Order
ASM              0
COBOL            0
COBOLE           0
COBOL2           0
JAVA             0
PLI              0
PLIE             0
SQL              0
***** Bottom of data *****

```

3. Exit the **application - Language Names** panel and save your changes.
4. From anywhere in the ChangeMan ZMF ISPF client, type **=A.A.4** on the **Command** or **Option** line and press **Enter**. The **application - Compile Procedures** panel is displayed.
5. Follow the instructions in the *ChangeMan ZMF Administrator Guide* to copy the global compile procedure for SQL started procedures into the **application - Compile Procedures** panel.

```

CMNCLPRC          ACTP - Compile Procedures      Row 1 to 10 of 10
Command ==> _____ Scroll ==> CSR

Language Procedure Description Order
ASM      CMNASM   Stage Assembler Source      0
C        CMNCEE   C build procedure            0
COBOLE   CMNCOBE   Stage IBM Enterprise COBOL source 0
COBOL2   CMNCOB2   COBOL2 source                0
COBOL2   CMNCO2OB   COBOL2 source to object       0
JAVA     CMNJAR    Create Java archive           0
JAVA     CMNJAVA   Stage Java source             0
PLI      CMNPLI   Stage PL/I Source             0
PLIE     CMNPLIE   Stage Enterprise PL/I source   0
SQL      CMNSQL    Translate, compile, and link SQL Stored Proc 0
***** Bottom of data *****

```

6. Exit the **application - Compile Procedures** panel and save your changes.

## Add Baseline Libraries for Db2 Components

Add baseline libraries for each library type you added for Db2 components in this application.

1. From anywhere in the ChangeMan ZMF ISPF client, type **=A.A.B** on the **Command** or **Option** line and press **Enter**. The **application - Baseline Configuration Part 1 of 2** panel is displayed.
2. Follow the instructions in the *ChangeMan ZMF Administrator Guide* to insert a baseline library type row and specify a baseline library description for each kind of Db2 component you will manage in this application.

See the baseline configuration for Db2 library types in this example:

Type	Levels	Install in prod	Baseline storage means
CPY	10	N	SD
CP2	10	N	SD
CTC	10	Y	SD
DBB	10	Y	SD
DBR	3	Y	SD
DOC	10	N	SD
HTH	3	N	H
JAR	3	Y	H
JCF	3	N	H
JCL	10	Y	SD
JCT	3	N	H
JVL	2	Y	H
JVS	2	Y	H
JVT	2	Y	H
LCT	10	N	SD
LOD	3	Y	P
LOS	3	N	P
LSH	3	N	H
LST	3	N	P
OBJ	10	N	SD
PKG	10	Y	SD
PRC	10	Y	SD
SPD	10	Y	SD
SPN	10	Y	SD
SPQ	10	Y	SD
SRC	10	N	SD
SRS	10	N	SD
STL	3	Y	P
STP	10	N	SD
TRG	10	Y	SD
TST	10	N	SD
UDF	10	Y	SD
WAR	3	Y	H
WCT	3	N	H

\*\*\*\*\* Bottom of data \*\*\*\*\*

3. Type **S** on the Line Command for each new library type row to display the **application - Baseline Configuration Part 2 of 2** panel.

- a. Follow the instructions in the *ChangeMan ZMF Administrator Guide* to verify existing libraries that you will use as baseline libraries or to allocate new libraries.
- b. Ensure that the **Data Set Type** field for libraries containing stored procedure load modules is **LIBRARY** (PDSE).

See the baseline libraries for Db2 library type DBB in this example:

```

CMNCBAS2      ACTP - Baseline Configuration Part 2 of 2      Row 1 to 2 of 2
Command ==> _____ Scroll ==> CSR

Library type:      DBB
Levels maintained: 10
Storage means:     Stacked Reverse Delta

Lvl Dataset name  +      Status
0  CMNTP.S6.V810.BASE.ACTP.DBB      *Verified
009 CMNTP.S6.V810.BASE.ACTP.DBB.DELTA      *Verified
***** Bottom of data *****


```

4. Exit the **application - Baseline Configuration** panels and save your changes.

## Add Production Libraries for Db2 Components

If you specified **Y** in **Install in Production** in a baseline configuration for a Db2 component type, you must define production libraries for that library type.

1. From anywhere in the ChangeMan ZMF ISPF client, type **=A.A.P** on the **Command** or **Option** line and press **Enter**. The **application - Production Libraries** panel CMNCPRL is displayed.
2. Follow the instructions in the *ChangeMan ZMF Administrator Guide* to insert a production library type and specify a set of production libraries for each type.

 **Note**

Temporary installs for stored procedures, user defined functions, and triggers are not supported. For these library types, type NULLFILE in the second production library line.

See the production libraries in this example for one Db2 bind control library type and one user defined function library type.

```

CMNCPRL      ACTP - SERT6 Production Libraries      Row 8 to 16 of 16
Command ==> _____ Scroll ==> CSR

Type Production dataset name  +
Temporary dataset name      +
Backup dataset name          +
DBB CMNTP.S6.V810.PROD.DBB
CMNTP.S6.V810.PROD.DBB.TEMP
CMNTP.S6.V810.PROD.DBB.BKUP
...
UDF CMNTP.S6.V810.PROD.UDF
NULLFILE
CMNTP.S6.V818.PROD.UDF.BKUP
***** Bottom of data *****

```

3. Exit the **application - Production Libraries** panel and save your changes.

# Configure Db2 Option Global Administration

Global Administration for the ChangeMan ZMF Db2 Option defines:

- Physical Db2 subsystems that are available to the Db2 Option.
- Logical Db2 subsystems that are available to Db2 Application Administration to define automated processing for Db2 components at promotion and install.
- Library types that are available to Db2 Application Administration to define special Db2 component processing.
- Connectors that define the relationship between a source and a target logical subsystem.
- General parameters that are available to Db2 Application Administrators to set options for processing Db2 components.

Type **=A.G.O.2** on any Command or Option line and press **Enter** to display the **Db2 Administration Options** panel:

```

CMNGDB2M          Db2 Administration Options          Row 1 to 28 of 28
Option ==> _____ Scroll ==> CSR

1 Physical Identify Db2 physical subsystems
2 Logical Define Db2 logical subsystems
3 Libtypes Define Db2 library type options
4 Connector Define source/target connector
G General Specify Db2 general parameters
  
```

This table describes the options on the **Db2 Administration Options** panel:

Option	Explanation
Physical	Identify Db2 subsystems and define JOB information and Db2 load libraries for Db2 Option jobs.
Logical	Define rules for modifying BIND PLAN and BIND PACKAGE commands at promotion or install. Define special processing for stored procedures and triggers.
Libtype	Set Db2 Sub Types to invoke special processing for library types that manage Db2 components.
Connector	Define the relationship between a source and a target logical subsystem. This is used for the BIND DEPLOY mechanism for distributing Native SQL stored procedures.

Option	Explanation
General	Specify options for processing Db2 components at stage and recompile.

## Define Physical Subsystems

Identify the Db2 subsystems where the ChangeMan ZMF Db2 Option executes functions.

### Note

If you license the ChangeMan ZMF ERO Option, you must also identify the Db2 subsystem where the ERO Db2 tables are defined.

1. On the **Db2 Administration Option** panel, choose option 1 Physical and press **Enter**. The **Db2 Physical Subsystems - Part 1 of 2** panel CMNGD2S0 is displayed:

```

CMNGD2S0          Db2 Physical Subsystems - Part 1 of 2          Row 1 to 4 of 4
Option ===> _____ Scroll ===> CSR

      Db2
      subsys  Site          Db2 System Load Library
      C105    SERT6         SYS2.DB2810.SDSNLOAD
      C105    SERT6P1       SYS2.DB2810.SDSNLOAD
      C105    SERT6P2       SYS2.DB2810.SDSNLOAD
      C105    SERT6P2       SYS2.DB2810.SDSNLOAD
***** Bottom of data *****

```

This table describes fields on the **Db2 Physical Subsystems - Part 1 of 2** panel:

Field	Description
Line Command	Type a line command to the left of a panel row:
	<b>S</b> Display the <b>Db2 Physical Subsystems - Part 2 of 2</b> panel for this physical subsystem.
	<b>I</b> Insert a blank row.
	<b>R</b> Repeat an existing row. <b>D</b> Delete an existing row.
	<b>D</b> Delete an existing row.
Db2 subsys	Type the Db2 subsystem identifier.
Site	Type the site where this Db2 subsystem runs. Your entry is validated against sites defined in global administration of the base ChangeMan ZMF product.

Field	Description
	Type * to see the <b>Global Site Selection List</b> panel.
	This entry must be blank for the first row, and this must reference the physical Db2 subsystem which houses the tables CMNx.CMNADMIN_NAMED and CMNx.CMNADMIN_GENERAL.
Db2 System Load Library	Type the data set name of the Db2 system load library that is used for this Db2 subsystem. You may leave this field blank if you LINKLIST the Db2 system load library.
	Also please note that the skeleton CMN\$\$\$D2X by default will use this name to build the SDSNEXIT dataset name. See the notes in the skeleton for more information.

### Important

If you license the ChangeMan ZMF Db2 Option, it is a requirement that the first row on this panel must identify a local Db2 subsystem where the Db2 Option tables are defined. The SITE field for the first physical subsystem definition on this panel must be blank. This is the physical Db2 subsystem which the ZMF started task connects to (via CAF connect) to access Db2 tables required to support the ZMF Db2 Option.

If you license the ChangeMan ZMF ERO Option, the ERO Db2 tables must also reside on the first physical Db2 subsystem as described above.

Define a physical Db2 subsystem for every Db2 subsystem where you want ChangeMan ZMF to manage Db2 components.

2. On the **Db2 Physical Subsystems - Part 1 of 2** panel, type **S** on the Line Command for a physical subsystem row and press **Enter**. The **Db2 Physical Subsystems - Part 2 of 2** panel CMNGD2S1 is displayed:

```

CMNGD2S1          Db2 Physical Subsystems - Part 2 of 2
Command ==> _____

Db2 subsystem: C105
Site:          SERT6
Load Library:  SYS2.DB2810.SDSNLOAD

Job statement information for Db2 binds:
//SERT6DB JOB (X170,374),SERT6,
// CLASS=A,MSGCLASS=X
//*
//*
```

This table describes fields on the **Db2 Physical Subsystems - Part 2 of 2** panel:

Field	Description
Db2 Subsystem	Displays the Db2 subsystem identifier.
Site	Displays the site where this Db2 subsystem runs.
Load library	Displays the data set name of the Db2 system load library that is used for this Db2 subsystem.
Job statement information for Db2 binds	Type JOB statement information for batch jobs that perform Db2 Option functions in the Db2 subsystem for promote and install.

Type Job Statement Information for every physical Db2 subsystem.

## Define Global Logical Subsystems

Configure logical Db2 subsystems that define automated processing for Db2 components at promotion and install.

1. On the **Db2 Administration Option** panel, choose option **2 Logical** and press **Enter** to display the **Db2 Logical Subsystems** panel:

```

CMNGD2LN          Db2 Logical Subsystems - Part 1 of 2          Row 1 to 3 of 3
Option ===> _____ Scroll ===> CSR

Line commands:
  P   Specify miscellaneous parameters
  T B Bind plan/pkg process  named(T) and general(B) templates
  Q G SQL process           named(Q) and general(G) templates
  V H Bind service process  named(V) and general(H) templates

Logical   Db2
name      subsys  Site      Description
SERT7     Q10K    SERT7     SERT7 D/P INSTANCE
SERT7P1   Q10K    SERT7     SERT7 PROMOTION SITE \#1
SERT7P2   Q10K    SERT7     SERT7 PROMOTION SITE \#2
***** Bottom of data *****

```

This table describes fields on the **Db2 Logical Subsystems** panel:

Field	Description
Line Command	Type a line command to the left of a panel row.
	I Insert a blank row.
	R Repeat an existing row.



Field	Description
	<b>D</b> Delete an existing row.
	<b>P</b> Specify miscellaneous processing parameters.
	<b>T</b> Specify BIND command named parameter templates.
	<b>B</b> Specify BIND command general token templates.
	<b>Q</b> Specify SQL process named variable templates.
	<b>G</b> Specify SQL process general token variable templates.
	<b>V</b> Specify BIND SERVICE command named parameter templates.
	<b>H</b> Specify BIND SERVICE command general token templates.
Logical name	Type a 1-8 character mnemonic for this logical Db2 subsystem. Db2 logical subsystem names must be unique across all physical subsystems. The Logical Name is also called Db2 nickname in this manual.
Db2 subsys	Type the Db2 physical subsystem where the parameters and templates in this logical subsystem will be used. Type * to display the Db2 Physical Subsystem List to select valid Db2 Subsys and Site combinations.
Site	Type the site where the Db2 physical subsystem runs. Type * to display the Db2 Physical Subsystem List to select valid Db2 Subsys and Site combinations.
Description	Type a 30-character description for the logical subsystem.

Create a logical subsystem for every promotion level and production environment where the Db2 Option will manage Db2 components.

2. On the **Db2 Logical Subsystems** panel, type **P** on the Line Command for a logical subsystem row and press **Enter**. The **Db2 Logical Subsystem nickname Parameter Settings** panel CMNGD2PM is displayed:

```

CMNGD2PM          Db2 Logical Subsystems SERT6 Settings
Command ==> _____

Preferred Libtypes:
DBRM . . . . .
Plan bind parameters . . . . .
Package bind parameters . . . . .
Service grants . . . . .

General Parameters:

Enter "/" to select option
  Bind Failure is significant
  Recycle Stored Procedures where WLM Environment is . .
  Maintain Trigger Sequence
  Use Db2 versioning for Native SQL Stored Procedures

```

This table describes fields on the **Db2 Logical Subsystem nickname Parameter Settings** panel:

Field	Description
Preferred Libtypes	<p>These fields are not used unless:</p> <ol style="list-style-type: none"> <li>1. You assign Db2 subtypes B (BIND PLAN) or P (BIND PACKAGE) or R (DBRM) to more than one library type in this application. See <a href="#">Define Global Db2 Library Subtypes</a>.</li> <li>2. You customize promotion, demotion, and installation skeletons to use the library types entered in these fields. The data in these fields is available in ISPF variables NTDBR, NTDBB and NTDBP in tables CMNDB2NN and CMNDB2N1.</li> <li>3. The service grant libtype is only used when automating BIND SERVICE commands.</li> </ol>
Bind Failure is significant	Select this to stop promote or demote processing if a Db2 bind fails in this logical subsystem. Leave this blank to continue promote or demote processing if a bind fails in this logical subsystem.
Recycle Stored Procedures	Select this to issue Db2 command VARY WLM...REFRESH to refresh a stored procedure or external user defined function that has changed in this logical subsystem. If not selected, then do not automatically refresh a stored procedure or external user defined function that has changed in this logical subsystem.

Field	Description
Where WLM Environment Is	If stored procedures are executed in one or more WLM-managed address spaces, type the name (or pattern) for the target WLM environment. The value of this field restricts the refresh of stored procedures to those environments that match the name or pattern you specify. You can wildcard this field by typing an asterisk at the end to specify a pattern for matching WLM environments. For example, C102* targets all WLM-managed environments whose names begin with the characters C102.
Maintain Trigger Sequence	Select this to drop and recreate all triggers for an event/table combination when one trigger is changed. Triggers are ordered according to the first 10 characters in the COMMENT ON field in the CREATE TRIGGER SQL. If not selected, then do not drop and recreate other triggers for an event/table combination when one trigger is changed. The modified trigger will execute last.
Use Db2 versioning for Native SQL Stored Procedures	Select this to use Db2 versioning for Native SQL stored procedures. If not selected, then drop/create will be used for this logical subsystem

Set parameters, then press **Enter** to accept panel entries.

#### **Note**

The entries on this panel do not restrict entries on the **Db2 Logical Subsystem nickname Settings** panel at the application level. The entries at the global level provide a model for applications.

3. On the **Db2 Logical Subsystems** panel, type **T** on the Line Command for a logical subsystem row and press **Enter**. The **Db2 Logical Subsystem nickname BIND Process Templates** panel CMNGD2L2 is displayed:

```

CMNGD2L2      Db2 Logical Subsystem PROD BIND Process Templates
Command ==>>> _____

  Templates      Target          Source          Insert
General:
Qualifier . . . .      +              +              +
Owner . . . . .      +              +              +

Plan:
Name . . . . .

Package:
Location . . . .      +              +


Collection . . . .      +              +

```

This panel defines BIND command templating that is performed for this Db2 logical subsystem.

How you use the fields on this panel to achieve the templating that you need is explained by example in [Templating Examples](#). For an introduction to templating, see [Templates](#).

The two tables that follow explain the field names at the left of the panel and the templating names at the top of the panel.

 **Note**

All data fields on this panel, except for Plan Name, exceed the length of the displayed panel fields. See topic "Working with Long Fields" in the *ZMF User's Guide* for instructions on how to enter, update, and erase data in long panel fields.

This first table defines the field names at the left of the **Db2 Logical Subsystem nickname BIND Process Templates** panel.

Syntax of the BIND PLAN and BIND PACKAGE commands referred to in this table:

```

BIND PLAN(plan-name) PKLIST(location-name.collection-id.package-id)
  - OWNER(authorization-id) QUALIFIER(qualifier-name)...

BIND PACKAGE(location-name.collection-id)
  - OWNER(authorization-id) QUALIFIER(qualifier-name)...

```

Field	Description
Qualifier	Template or insert value for qualifier-name in BIND PLAN commands and BIND PACKAGE commands. Qualifier may be up to 128 characters long.
Bind Owner	Template or insert value for authorization-id in BIND PLAN and BIND PACKAGE commands. Bind Owner may be up to 128 characters long.


Field	Description
PLAN Name	Template for plan-name in BIND PLAN commands. Plan name may be up to 8 characters long.
PACKAGE Location	Template for location-name in BIND PACKAGE commands. If the PKLTEMPLATE control statement is input to plan lookup program CMNDB2PL, then the template is also applied to the location-name in the PKLIST parameter of BIND PLAN commands. See the "PKLTEMPLATE" table entry in <a href="#">Keyword Control Statements</a> . Package Location may be up to 128 characters long.
PACKAGE Collection	Template for collection-id in BIND PACKAGE commands. If the PKLTEMPLATE control statement is input to plan lookup program CMNDB2PL, then the template is also applied to the collection-id in the PKLIST parameter of BIND PLAN commands. See the "PKLTEMPLATE" table entry in <a href="#">Keyword Control Statements</a> . Package Collection may be up to 128 characters long.

This second table defines templating fields **Target**, **Source**, and **Insert** on the **Db2 Logical Subsystem nickname BIND Process Templates** panel in terms of the kind of templating that is performed.

Template Type	Field	Description
Replace characters at an offset	Target	Placeholder ? characters define the offset for replacement characters. Example: ???S?T replaces the fourth character of a seven-character value with S and the sixth character with T.
	Source	Blank
	Insert	Blank
Add characters at end	Target	Placeholder ? characters define a field that is as long or longer than the actual data, followed by characters to be appended to the parameter value. Example: ???S?T adds ST to the end of a three-character value.
	Source	Blank
	Insert	Blank

Template Type	Field	Description
Replace characters at end	Target	Character * (asterisk) indicates the start of a literal string n characters long that will replace the last n characters of the parameter value. Example: *ST replaces the last two characters with ST.
	Source	Blank
	Insert	Blank
Delete characters at end	Target	Character ~ (not) indicates a field character that will be replaced with a space. Since embedded spaces are invalid in a parameter value, use ~ to delete characters at the end of a value. Example: ?????~~ deletes the last two characters of a six character value or the last character of a five character value.
	Source	Blank
	Insert	Blank
Replace character string	Target	Literal string that will replace the first occurrence of the string matching the value in the <b>Source</b> field. The matching string and replacing string can be different lengths.
	Source	Literal string to search for.
	Insert	Blank
Add an OWNER parameter	Target	Blank
	Source	Blank
	Insert	Value for the OWNER parameter.
		<b>Note:</b> There must be no OWNER in the input BIND command, and the following control statement must be input to the plan lookup program CMNDB2PL: AUTHORITY=OWNER,INSERT
Add a QUALIFIER parameter	Target	

Template Type	Field	Description
	Source	Blank
	Insert	Value for the QUALIFIER parameter.
<p><b>Note:</b> There must be no QUALIFIER in the input BIND command, and the following control statement must be input to the plan lookup program CMNDB2PL: INSERTQUAL</p>		

 **Note**

The entries on this panel do not restrict entries on the **Db2 Logical Subsystem nickname Templates** panel at the application level. The entries at the global level provide a model for applications.

4. On the Db2 Logical Subsystems panel, type B on the line command for a logical subsystem row and press ENTER.

The Db2 Logical Subsystem nickname BIND General Templates panel CMNGD2L5 is displayed. From this panel, the BIND process general tokens for that particular logical subsystem can be specified.

 **Note**

This facility is not part of the Core Db2 option. Only named templates are available if you choose to use the Core option.

```

CMNGD2   Db2 Logical Subsystem SERT7 BIND General Templates Row 1 to 21 of 21
Command ===> _____ Scroll ===> CSR

Token name      + Target template      + Source template      +
CURRENTSERVER   ???DSNP                    _____
DEGREE          0                            _____
EXPLAIN         NO-                          _____
PATH            >REMOVE<                    EPICPYYY
PATH            >REMOVE<                    EPICPXXX
PATH            ???P???                    _____
PKLIST          CA_PRD                      CA_TNG
_____
_____

***** Bottom of data *****

```

You can use I, R, D line commands to insert, repeat and delete rows in the table.

Each row in this table represents a BIND process general token template. The token name must match the particular bind parameter you wish to change. The target and source templates work in exactly the same way as the standard 'named' templates. The new facility within this templating process is the use of the '>REMOVE\<' target template. Use of >REMOVE\< will exclude any sub-parameter in a list for the main parameter in question where any part of that sub-parameter value matches the source template. This is only relevant for parameters which support lists of values. In the above example any PATH values which match either EPICPYYY or EPICPXXX will be removed from the list.

You can define templating for any bind parameter you wish using these general 'token' templates. The bind command is parsed into a distinct set of parameters and associated values. The general token template 'name' will be matched against these parameters. The value associated with the parameter will be transformed by the template.

If a parameter is associated with a list of subparameter values (e.g. PKLIST etc) then the templates will be applied to each value in turn. If you wish to remove a value from a list you can specify >REMOVE\< as the target template and, if the source template matches one or more subparameter values in the list, then those values will be removed from the list.

There are three kinds of transformations available.

You can unconditionally override a value by using a blank source template together with a non-blank target template. The target template can contain a mixture of wildcard place holders (?) and constant literals. It can also contain the logical not character (-) which will cause a blank to replace the relevant position in the target string. If a template containing wildcards is longer than the name being transformed then the wildcards are 'squeezed' from the right. The target name is then overlaid with the specified constant literals.

If the template consists of a string of place holders followed by one or more literals, and the string is longer than the target name, then the literals will be appended to the target name.

Alternatively, you can use a blank source template and specify \* as the first character of the target template. This will cause the 'n' characters following the \* to replace the last 'n' characters of the target name. Note that the wildcard placeholder (?) has no special meaning in this kind of transformation but that - still represents a blank character override.

For example, \*QA applied to VAL01 and TKNVAL01 will result in VALQA and TKNVALQA respectively.

You can also conditionally search for strings to be replaced. To do this specify the search string in the source template field. Specify the replace string in the target template field. Neither wildcard character (? or \*) has any special meaning in this kind of transformation but the - character still represents a blank override. If the replacement string is shorter than the search string then the rest of the name is 'shuffled up' as appropriate. If the replacement string is longer and this results in field length overflow, then the rightmost characters will be lost.

The rules for the token 'name' and how it is searched for follow.

The token 'name' is up to 64 bytes in length and can consist of up to 5 words, each no longer than 16 bytes. If the token name contains imbedded blanks then it must be enclosed in single



quotes. This is to fit in with the same facility used in SQL processing. However, unlike SQL general token templating where we are scanning a complete SQL sentence, here we are attempting to match the general token name with a specific bind parameter (e.g. PATH, EXPLAIN etc.).

For example:

```
Token Name = EXPLAIN
Target Template = NO-
Source Template =
```

This will scan for the presence of the EXPLAIN parameter and will replace its value with NO (note the not sign is required to avoid replacing YES with NOS in this case).

5. On the **Db2 Logical Subsystems** panel, type **Q** on the Line Command for a logical subsystem row and press **Enter**. The **Db2 Logical Subsystem nickname SQL Process Templates (named)** panel CMNGD2L3 is displayed:

```
CMNGD2L3      Db2 Logical Subsystem QAD1 SQL Process Templates (Named)
Command ==> _____
```

Templates	Target	Source	Deploy
Schema . . . .	_____ +	_____ +	+
Collection . .	_____ +	_____ +	+
WLM . . . . .	_____ +	_____ +	+
Location . . .	_____ +	_____ +	_____ +
Qualifier . . .	_____ +	_____ +	_____ +
Owner . . . . .	_____ +	_____ +	_____ +

This panel defines SQL templating that is performed for this Db2 logical subsystem. How you use the fields on this panel to achieve the templating that you need is explained by example in [Templating Examples](#). For an introduction to templating, see [Templates](#). The two tables that follow explain the field names at the left of the panel and the templating names at the top of the panel.

!!!! note All data fields on this panel exceed the length of the displayed panel fields. See topic "Working with Long Fields" in the ZMF User's Guide for instructions on how to enter, update, and erase data in long panel fields.

This first table defines the field names at the left of the **Db2 Logical Subsystem nickname SQL Process Templates (Named)** panel:

Syntax of the CREATE (and other DDL) commands referred to in this table:

```

...
CREATE PROCEDURE schema.procedure-name...COLLID collection-id...
CREATE FUNCTION schema.function-name...COLLID collection-id...
CREATE TRIGGER schema.trigger-name...ON qualifier-name.table-name...
...

```

Field	Description
Schema	Template for explicit schema in procedure-name, functionname, or trigger-name in DDL. Schema may be up to 128 characters long.
WLM	Template to be applied to any WLM ENVIRONMENT clause found in the DDL for (e.g.) an external stored procedure.
Location	Template applied to a location identifier in DDL (see more about this in the DEPLOY sections below).
Qualifier	Template applied to DDL qualifier clauses
Owner	Template applied to DDL Package Owner clauses

\*\*Deploy Fields | Field | Description | |-----|-----| | LOCATION | This value is used to route the BIND DEPLOY command to the relevant source Db2 subsystem. Its value is picked up from the source logical subsystem of the two tied together by the CONNECTOR definition (more below) | | QUALIFIER | If the usual templates do not generate a non-blank qualifier then whatever is coded here will be used in the QUALIFIER clause of the generated BIND DEPLOY command | | OWNER | If the usual templates do not generate a non-blank owner then whatever is coded here will be used in the OWNER clause of the generated BIND DEPLOY command |

6. On the **Db2 Logical Subsystems** panel, type **G** on the Line Command for a logical subsystem row and press **Enter**. The **Db2 Logical Subsystem nickname SQL Process Templates (general)** panel CMNGD2L4 is displayed:

## CMNGD2L4

```

CMNGD2L4 B2 Logical Subsys PROD SQL Process Templates (Ge Row 1 to 21 of 21
Command ===> _____ Scroll ===>

Token name          Target template      Source template
-----+-----+-----+
-----+-----+-----+
-----+-----+-----+
-----+-----+-----+
...
-----+-----+-----+
-----+-----+-----+
***** Bottom of data *****

```

This panel defines SQL templating that is performed for this Db2 logical subsystem.

You can define templating for any keyword you wish using these general 'token' templates.

```

ZMF will look for your keyword, i.e. the token name and will then apply the templates to
the value associated with that keyword.

```

The token 'name' is up to 64 bytes in length and can consist of up to 5 words, each no longer than 16 bytes. If the token name contains imbedded blanks then it must be enclosed in single quotes. ZMF will scan the SQL/DDDL, squeezing white space, looking for the token words terminated by either a blank or left parenthesis. The value of the word following on from our token name will be templated as requested.

For example:

```

Token Name = 'SYSTEM TIME SENSITIVE'
Target Template = YES
Source Template =

```

This will scan for the presence of SYSTEM TIME SENSITIVE in the SQL and will change any value following this (e.g. NO) to YES.

1. On the **Db2 Logical Subsystems** panel, type **V** on the Line Command for a logical subsystem row and press **Enter**. The **Db2 Logical Subsystem *nickname* Bind Service Named Templates** panel CMNGD2L6 is displayed:

```
CMNGD2L6 Db2 Logical Subsystem QAD1 Bind Service Named Templates
Command ===>
```

Templates	Target	Source
Collection . . .	_____	+ _____ +
Qualifier . . .	_____	+ _____ +
Owner . . . . .	_____	+ _____ +

These three named parameters are templated in the same way as for package/plan binds (see section 3 above). The underlying processes are different.

- On the Db2 Logical Subsystems panel, type **H** on the line command for a logical subsystem row and press **ENTER**. The **Db2 Logical Subsystem *nickname* BIND Service General Templates** panel CMNGD2L7 is displayed, from which the BIND Service process general tokens for that particular logical subsystem can be specified:

```
CMNGD2L7 Db2 Logical Subsystem QAD1 Bind Service General Temp Row 1 to 13 of 13
Command ===> Scroll ===> CSR
```

Token name	+ Target template	+ Source template	+
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	

This general token processing is described in step 4 above. However, there are some extra features which are applicable only to Bind Service processing.

When a grant SQL component is associated with the bind service process and the token name is >GRANTEE< , the template is applied to the list of grantee userids/groups on the grant SQL supplied to the process.

Also, if you enclose the source template in single quotes, the target template is only used if the value to be replaced exactly matches the source template value.

## Define Global Db2 Library Subtypes

Db2 library subtypes invoke special processing for Db2 components. When you defined the Global Library Types for these components, you coded D in the Selectable Option field. Here you assign a Db2 Sub Type to each of those library types.

On the **Db2 Administration Option** panel, choose option **3** Libtypes and press **Enter** to display the **Db2 Library Types** panel:

```

CMNDGLT0                               Db2 Library Types                               Row 1 to 11 of 11
Command ==> _____ Scroll ==> CSR

Lib                                     Db2
type      Description                  sub   End SQL
PKG       Db2 Bind Package Commands   P
PRC       Cataloged Procedures
DBB       Db2 BIND PLAN Commands      B
DBR       Db2 DBRM                     R
SPD       Db2 Stored Proc Definitions - Non-SQL D
SPN       Db2 Stored Proc Source - Native SQL D      @
SPQ       Db2 Stored Proc Source - SQL Language Q      @
STL       Db2 Stored Proc Load Modules S
STP       Db2 Stored Proc Source - External Lan
TRG       Db2 Trigger Definitions      T
UDF       Db2 User-Defined Function Definitions D
***** Bottom of data *****

```

This table describes the fields on the **Db2 Library Types** panel:

Field	Description
Lib type	Displays the library types from the Global Library Type definition that are defined with a Selectable Option of D.
Description	Displays the definition from the global library type.
Db2 sub type	Type the Db2 Sub Type for special Db2 Option processing. Sub type processing is described in the next table below.
	<b>B</b> BIND PLAN command
	<b>D</b> CREATE statements for stored procedures, external user defined functions.
	<b>N</b> Native SQL Stored Procedure definition
	<b>Q</b> BIND PACKAGE command SQL stored procedure source
	<b>R</b> DBRM Stored procedure load modules, REXX stored procedures
	<b>T</b> Trigger definition source
	<b>V</b> BIND SERVICE command or associated GRANT DDL
End SQL sentence	Type an alternate SQL statement terminator. If the components in this library type include SQL that uses the semicolon (;) as a statement terminator, specify an alternate terminator for the stored procedure or function so that the semicolon is passed through to the server. You can specify any character except the following:

Field	Description
	- comma
	- underscore
	- single quote
	- double quote
	- left hand parenthesis
	- right hand parenthesis
	If you leave this field blank, the default alternate SQL statement terminator is semicolon (;).

Define sub types, then press **Enter** to accept panel entries.

#### Note

The entries on this panel do not restrict entries on the Db2 Library Types at the application level. The entries at the global level provide a model for applications.

This table shows the processing assigned by Db2 Option library sub types:

Sub Type	Description	Modified Process	Sub Type Processing Description
B	BIND PLAN command.	Promote Demote Install Backout	Process with plan lookup program CMNDB2PL to template BIND parameters for the target Db2 subsystem.
D	CREATE statements for stored procedures, external user defined functions.	Promote Demote Install Backout	Process with utility program CMNDB2DD to register the object in the Db2 catalog. Issue a DROP before the CREATE.
N	Native SQL stored procedure definitions.	Promote Demote Install Backout	Drop/Create, Alter add version, and bind deploy mechanisms supported for this library type.

<b>Sub Type</b>	<b>Description</b>	<b>Modified Process</b>	<b>Sub Type Processing Description</b>
P	BIND PACKAGE command.	Promote Demote Install Backout	Process with plan lookup program CMNDB2PL to find applicable BIND PLAN members and template parameters for the target Db2 subsystem.
Q	SQL stored procedure source.	Promote Demote Install Backout	Process with utility program CMNDB2DQ to remove SQL procedural code, then process the CREATE with utility program CMNDB2DD to register the stored procedure in the Db2 catalog.
R	DBRM		Process with plan lookup program CMNDB2PL to find applicable BIND PACKAGE and BIND PLAN members and template BIND parameters for the target Db2 subsystem.
S	Stored procedure load modules, REXX stored procedures.	Promote Demote Install Backout	If the Recycle Stored Procedure field is YES, issue Db2 commands VARY WLM...REFRESH in the WLM-managed address space to refresh the executable.
T	Trigger definition source.	Promote Demote Install Backout	Process with utility program CMNDB2DD to extract the table/event/time combinations. If Maintain Trigger Sequence YES, query SYSIBM.SYSTRIGGERS with utility program CMNDB2TR to find multiple triggers defined for the same table/event/time combination, then drop and recreate those triggers to maintain the original firing order.

<b>Sub Type</b>	<b>Description</b>	<b>Modified Process</b>	<b>Sub Type Processing Description</b>
V	BIND SERVICE command and associated GRANT SQL	Promote Demote Install Backout	Process with CMNDB2SV to apply templates to incoming BIND SERVICE and GRANT components. The templated BIND SERVICE command is passed to an IKJEFT01 step for processing. Any templated GRANT DDL is passed to CMNDB2GR for action.

This table shows you how the Db2 Option sub types relate to library types and other Db2 Option parameters. Sub types are shown in **bold**.

<b>Db2 Component</b>	<b>Like</b>	<b>Target Type</b>	<b>Sel Opt</b>	<b>Sub Typ</b>	<b>BIND/SQL/SERVICE</b>
Db2 Application Program Source	S	Db2 Program Load			
Db2 Application Program Load	L				B
DBRM	P		D	**R	B
BIND PLAN Command	P		D	**B	B
BIND PACKAGE Command	P		D	**P	B
External Stored Procedure Source	S	Stored Procedure Load			
External SQL Stored Procedure Source	S	Stored Procedure Load	D	**Q	S
External Stored Procedure Load	L		D	**S	B & S
Native SQL Stored Procedure	P		D	**N	S



Db2 Component	Like	Target Type	Sel Opt	Sub Typ	BIND/SQL/SERVICE
General DDL (e.g. CREATE PROCEDURE for external SP)	P		D	**D	S
User Defined Function Definition	P		D	**D	S
Trigger Definition	P		D	**T	S
BIND Service command	P		D	V	V
Service GRANT command	P		D	V	V

\* Db2 Active Library specification for BIND plan/pkg (B), Process SQL (S), and Bind Service (V).

## Define Source/Target Connector

On the **Db2 Administration Option** panel, choose option **4 Connector** and press **Enter** to display the **Logical Subsystem Connectors - Global List** panel (CMNGD2CL):

```

CMNGD2CL      Logical Subsystem Connectors - Global List      Row 1 to 11 of 11
Command ==> _____      Scroll ==> CSR

Connector      Source      Target      Description
name           name       name
DS2UNIT        STUDIO     UNITV       DATA STUDIO TO UNIT (DSN)
QA2PROD        QAD        PRODD       QAD TO PRODD
QA2PROD1       QAD        PRODD1      QAD TO PRODD1 (DSN1)
UNIT2PRD       UNITV      PRODV       UNIT TO PROD (DSN)
UNIT2PR1       UNITV      PRODV1      UNIT TO PROD (DSN1)
UNIT2QA        UNIT       QAD         UNIT TO QAD
UNIT2QA1       UNITV     QAD1        UNIT TO QAD1 (DSN1)
***** Bottom of data *****

```

### Db2 Logical Subsystem Connectors - Global List

The BIND DEPLOY mechanism for distributing Native SQL stored procedures requires both a source and a target Db2 environment. This panel is used to 'connect' a source logical subsystem to a target logical subsystem.

Both logical subsystem names must already exist (you can enter an asterisk in either source or target name fields to get a list).

Values from the source logical subsystem are used to identify the stored procedure which will be deployed. Values from the target logical subsystem are used to specify the name and related attributes of the stored procedure when it is deployed to the target.

If you wish to see which values will be used for your choice of source and target then select the row once both names have been entered.

 **Note**

If you change the source and/or target subsystem on a row then you must save the changes before the new values will be displayed using the 'S' line command.

To invoke the BIND DEPLOY mechanism based on a specific connector name make the connector name the subsystem name associated with the relevant SQL active library.

Field	Description
CONNECTOR NAME (Required)	Select a one to eight-character mnemonic for this connector. This name must be unique.
SOURCE NAME (Required)	This must be an existing logical subsystem name and specifies the values which will be used to identify the source of the BIND DEPLOY command.
TARGET NAME (Required)	This must be an existing logical subsystem name and specifies the values which will be used to specify the target of the BIND DEPLOY command.
DESCRIPTION	Use this field to describe the use intended for this connector.

Selecting the first of the connector definitions above shows the panel CMNGD2CN thus:

```

CMNGD2CN          Logical Subsystem Connector Global Model - DS2UNIT
Command ===> _____

          DATA STUDIO TO UNIT (DSN)

Source . . . . STUDIO
Subsystem id . DSN
Location . . . DB2V11          +

Templates      Target          Source
Schema . . . .          +          +

Target . . . . UNITV
Subsystem id . DSN
Site . . . . LOCALVER

Templates      Target          Source          Deploy
Collection . . UNIT~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ +          +
Qualifier . . .UNIT~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ +          + UNIT          +
Owner . . . . SERD~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ +          + SERD          +

```

## Specify Global Db2 General Parameters

Define general parameters that are available to Db2 Application Administrators to set options for processing Db2 components.

On the **Db2 Administration Option** panel, choose option **G** General and press **Enter** to display the **Global Db2 General Parameters** panel (CMNGDPM0):

```

CMNGDPM0          Global Db2 General Parameters
Command ===> _____

Enter "/" to select option
_ Use Core Db2 Option Functions Only
_ Use Package Name in Db2 PC version
_ Force Pkg Name in Db2 PC version

```

This table describes the options and values on the **Global Db2 General Parameters** panel.

Field	Description
Use Core Db2 Option Functions Only	<p>The full functionality of the ZMF Db2 option allows you to manage various components such as stored procedures, REST services etc. To do this you need to define ZMF Db2 option administration tables. i.e.</p> <p>CMNx.CMNADMIN_NAMED</p> <p>CMNx.CMNADMIN_GENERAL</p> <p>CMNZMF.CMNDB2_ATTRIBS However, if you only need to use the core functionality of the Db2 option, i.e. managing package and plan binds, then you can select this global parameter and the requirement for these tables to exist goes away. If, at some later date, you unselect this option without setting up the relevant tables and re-binding the Db2 option packages then you will encounter runtime SQL errors in ZMF admin and file tailoring functions.</p>
Use Package Name in Db2 PC version	<p>Type Y or N to set boundaries for entering the “Use Package Name in Db2 PC version” option in Db2 Application Administration. The “Use Package Name in Db2 PC version” option determines whether the VERSION field on the <b>Db2 Physical Subsystems</b> panel (CMNSTG18) is initialized to the package ID in stage and recompile.</p>
	<p>Select to restrict settings for “Use Package Name in Db2 PC version” in Db2 Application Administration to be selected. In all applications, the VERSION field on the <b>Db2 Physical Subsystems</b> panel is initialized to the package ID in stage and recompile, but the field can be changed or blanked out.</p>
	<p>Omit to allow selection or otherwise for this option in Db2 Application Administration.</p>
Force Pkg Name in Db2 PC version	<p>This field is used to set boundaries for entering the “Force Pkg Name in Db2 PC version” option in Db2 Application Administration. The “Force Pkg Name in Db2 PC version” option determines whether the Package ID <i>must</i> be used for the VERSION parameter of the Db2 precompiler on the <b>Db2 Physical Subsystems</b> panel (CMNSTG18) in stage and recompile. This option must be omitted if the “Use Package Name in Db2 PC version” is not selected.</p>
	<p>Select this to restrict settings for “Force Pkg Name in Db2 PC version” in Db2 Application Administration to be selected. In all applications, the VERSION field on the <b>Db2 Physical Subsystems</b> panel is set to the package ID in stage and recompile, and the field <i>cannot</i> be changed.</p>

Field	Description
	Omit to allow any entry for this option in Db2 Application Administration.

Type your choices for the general parameters, then press **Enter** to accept panel entries. Press **PF3** to return to the **Db2 Options Administration** panel.

## Configure Db2 Option Application Administration

Application Administration for the ChangeMan ZMF Db2 Option defines:

Logical Db2 subsystems that define automated processing for Db2 components at promotion and install.

Active libraries that invoke Db2 Option processes at promotion and install defined by logical subsystems.

A library type for members containing BIND PACKAGE commands.

Library types that invoke special Db2 component processing.

General parameters for processing Db2 components.

Type **=A.A.O.2** on any Command or Option line and press **Enter** to display the **application - Db2 Administration Options** panel:

```

CMNLDB2M          STEV - Db2 Administration Options
Option ===\>

1 Logical          Define application Db2 logical subsystems
2 Library          Define application Db2 active library information
3 Libtype          Define Db2 library type options
4 Connector        Define source/target logical subsystem connector
5 Secondary        Define secondary bind requirements
6 General          Define general Db2 parameters for this application

```

This table describes the options on the *application - Db2 Administration Options*:

Field	Explanation
Logical	Define rules for modifying BIND PLAN and BIND PACKAGE commands at promotion or install. Define special processing for stored procedures and triggers.
Library	Define active libraries that invoke Db2 Option processing at promotion and install as defined by logical subsystems.
Libtype	Set Db2 Sub Types to invoke special processing for library types that manage Db2 components.

Field	Explanation
Connector	Define source/target logical subsystem connector for bind deploy of native-SQL stored procedures.
Secondary	Associate secondary bind logical subsystems to the primary bind logical subsystem.
General	Specify the use of package ID in the VERSION parameter for the Db2 precompiler.

## Define Application Logical Subsystems

1. On the *application* - **Db2 Administration Options** panel, choose option **1 Logical** and press **Enter** to display the **Db2 Logical Subsystems** panel:

```

CMNLD2LN                               Db2 Logical Subsystems                               Row 1 to 3 of 3
Command ===> _____ Scroll ===> CSR

Line commands:
  P      Specify miscellaneous parameters
  T B    Bind plan/pkg process named(T) and general(B) templates
  Q G    SQL process named(Q) and general(G) templates
  V H    Bind service process named(V) and general(H) templates

      Logical      Db2
      name         subsys      Site          Description
      SERT6        C105        SERT6        SERT6 D/P INSTANCE
      SERT6P1      C105        SERT6        SERT6 PROMOTION SITE #1
      SERT6P2      C105        SERT6        SERT6 PROMOTION SITE #2
***** Bottom of data *****

```

This table describes fields on the **Db2 Logical Subsystems** panel:

Field	Explanation
Line Command	Type a line command to the left of a panel row.
	I Insert a blank row. R Repeat an existing row.
	D Delete an existing row.
	P Specify miscellaneous processing parameters
	T Specify BIND process named variable templates B Specify BIND process general token variable templates
	Q Specify SQL process named variable templates

Field	Explanation
	G Specify SQL process general token variable templates
	V Specify BIND SERVICE command named parameter templates
	H Specify BIND SERVICE command general token templates
Logical name	Type a 1-8 character mnemonic for this logical Db2 subsystem. Db2 logical subsystem names must be unique across all physical subsystems. The Logical Name is also called "Db2 <i>nickname</i> " in this manual.
Db2 subsys	Type the Db2 physical subsystem where the parameters and templates in this logical subsystem will be used.
Site	Type the site where the Db2 physical subsystem runs.
Description	Type a 30-character description for the logical subsystem.

2. On the **Db2 Logical Subsystem** panel, type **I** in the line command of a row below which you want to add a physical subsystem and press **Enter**.
3. Type **\*** in the line command for the new row, and the **Db2 Logical Subsystem List** panel is displayed.
4. Select a global Db2 logical subsystem from the **Db2 Logical Subsystem List** and press **Enter**. The logical subsystem is added to the **Db2 Logical Subsystem** panel.
5. On the **Db2 Logical Subsystems** panel, type **P** on the Line Command for a logical subsystem row and press **Enter**. The **Db2 Logical Subsystem nickname Settings** panel (CMNGD2PM) is displayed:

```

CMNGD2PM          Db2 Logical Subsystem SERT6 Settings
Command ==>>> _____

Preferred Libtypes:
DBRM . . . . .
Plan bind parameters . . . . .
Package bind parameters . . . . .
Service grants . . . . .

General Parameters:
Enter "/" to select option
_ Bind Failure is significant
/ Recycle Stored Procedures where WLM Environment is . .
/ Maintain Trigger Sequence
_ Use Db2 versioning for Native-SQL Stored Procedures

```

This panel defines BIND command templating that is performed for this Db2 logical subsystem.

This table describes fields on the **Db2 Logical Subsystem *nickname* Settings** panel:

<b>Field</b>	<b>Description</b>
Bind Failure is significant	Select to stop promote or demote processing if a Db2 bind fails in this logical subsystem.
	Omit to continue promote or demote processing if a bind fails in this logical subsystem.
Recycle Stored Procedures	Select to issue Db2 command VARY WLM...REFRESH to refresh a stored procedure or external user defined function that has changed in this logical subsystem.
	Omit this field to prevent automatically refreshing a stored procedure or external user defined function that has changed in this logical subsystem.
Where WLM Environment Is	If stored procedures are executed in one or more WLM-managed address spaces, type the name (or pattern) for the target WLM environment. The value of this field restricts the refresh of stored procedures to those environments that match the name or pattern you specify.
	You can wildcard this field by typing an asterisk at the end to specify a pattern for matching WLM environments. For example, C102* targets all WLM-managed environments whose names begin with the characters C102.
Use Db2 versioning for Native SQL Stored Procedures	Select this to use Db2 versioning for Native SQL stored procedures. Instructs ZMF to keep track of active versions during Native SQL Stored Procedure deployment. Resets the active version during demote/backout instead of re-presenting the prior version of the SP definition.
	If not selected, then do not use Db2 versioning for Native SQL stored procedures.



Field	Description
Maintain Trigger Sequence	Select to drop and recreate all triggers for an event/table combination when one trigger is changed. To use this facility you must arrange for the 'comment on' value for the trigger to start with CMNFIRE#nn. Triggers are then ordered according to the nn in this text. When more than one trigger is defined for a particular event/table combination, the triggers will 'fire' in the order in which they were created. When we change a trigger definition we drop and then recreate it thereby altering the execution sequence. If this parameter is set to '/' then we will drop and recreate, without change, all the other trigger definitions for this event/table combination in order to maintain the required firing sequence.
	Omit this field to not drop and recreate other triggers for an event/table combination when one trigger is changed. The modified trigger will execute last.
Preferred Libtypes	These fields are not used unless:
	1. You assign Db2 subtypes B (BIND PLAN) or P (BIND PACKAGE) or R (DBRM) to more than one library type in this application. See <a href="#">Define Application Db2 Library Subtypes</a>
	2. You customize promotion, demotion, and installation skeletons to use the library types entered in these fields. The data in these fields is available in ISPF variables NTDBR, NTDBB and NTDBP in tables CMNDB2NN and CMNDB2N.
	3. The service grant libtype is only used when automating BIND SERVICE commands.

Set parameters, then press **Enter** to accept panel entries.

#### Note

The entries on this panel are not restricted by entries on the Db2 Logical Subsystem *nickname* Parameter Settings at the global level. The entries at the global level are only a model for applications.

6. On the **Db2 Logical Subsystems** panel, type **T** on the Line Command for a logical subsystem row and press **Enter**. The **Db2 Logical Subsystem *nickname* Bind Process Templates** panel is displayed:

CMNGD2L2 Db2 Logical Subsystem PROD BIND Process Templates  
 Command ==> \_\_\_\_\_

Templates	Target	Source	Insert
General:			
Qualifier . . . . .	PROD~	+ _____	+ PROD +
Owner . . . . .	SERD~	+ _____	+ SERD +
Plan:			
Name . . . . .	_____	_____	
Package:			
Location . . . . .	_____	+ _____	+ _____
Collection . . . . .	_____	+ _____	+ _____

This panel defines BIND command and schema templating that is performed for this Db2 logical subsystem.

How you use the fields on this panel to achieve the templating that you need is explained by example in [Templating Examples](#). For an introduction to templating, see [Templates](#).

The two tables that follow explain the field names at the left of the panel and the templating names at the top of the panel.

 **Note**

All data fields on this panel, except for Plan Name, exceed the length of the displayed panel fields. See topic "Working with Long Fields" in the *ZMF User's Guide* for instructions on how to enter, update, and erase data in long panel fields.

This first table defines the field names at the left of the **Db2 Logical Subsystem *nickname* BIND Process Templates** panel:

Syntax of the BIND PLAN and BIND PACKAGE commands referred to in this table:

```
BIND PLAN(plan-name) PKLIST(location-name.collection-id.package-id)
  - OWNER(authorization-id) QUALIFIER(qualifier-name)...

BIND PACKAGE(location-name.collection-id)
  - OWNER(authorization-id) QUALIFIER(qualifier-name)...
```

Field	Description
Qualifier	Template or insert value for qualifier-name in BIND PLAN commands and BIND PACKAGE commands. Qualifier may be up to 128 characters long.
Bind Owner	Template or insert value for authorization-id in BIND PLAN and BIND PACKAGE commands. Bind Owner may be up to 128 characters long.

Field	Description
Plan Name	Template for plan-name in BIND PLAN commands. Plan name may be up to 8 characters long.
Package Location	Template for location-name in BIND PACKAGE commands. If the PKLTEMPLATE control statement is input to plan lookup program CMNDB2PL, then the template is also applied to the location-name in the PKLIST parameter of BIND PLAN commands. See the <b>PKLTEMPLATE</b> table entry in the <a href="#">Program Level Control Statements</a> section under <b>CMNDB2PL - BIND Utility</b> . Package Location may be up to 128 characters long.
Package Collection	Template for collection-id in BIND PACKAGE commands. If the PKLTEMPLATE control statement is input to plan lookup program CMNDB2PL, then the template is also applied to the collection-id in the PKLIST parameter of BIND PLAN commands. See the <b>PKLTEMPLATE</b> table entry in the <a href="#">Program Level Control Statements</a> section under <b>CMNDB2PL - BIND Utility</b> . Package Collection may be up to 128 characters long.

This second table defines templating fields **Target**, **Source**, and **Insert** on the **Db2 Logical Subsystem *nickname* BIND Process Templates** panel in terms of the kind of templating that is performed.

Template Type	Field	Description
Replace characters at an offset	Target	Placeholder ? characters define the offset for replacement characters.
		Example: ???S?T replaces the fourth character of a seven-character value with S and the sixth character with T.
	Source	Blank
	Insert	Blank
Add characters at end	Target	Placeholder ? characters define a field that is as long or longer than the actual data, followed by characters to be appended to the parameter value.
		Example: ???S?T adds ST to the end of a three-character value.

Template Type	Field	Description
	Source	Blank
	Insert	Blank
Replace characters at end	Target	Character * (asterisk) indicates the start of a literal string n characters long that will replace the last n characters of the parameter value.
		Example: *ST replaces the last two characters with ST.
	Source	Blank
	Insert	Blank
Delete characters at end	Target	Character ~ (not) indicates a field character that will be replaced with a space. Since embedded spaces are invalid in a parameter value, use ~ to delete characters at the end of a value.
		Example: ???~-- deletes the last two characters of a six character value or the last character of a five character value.
	Source	Blank
	Insert	Blank
Replace character string	Target	Literal string that will replace the first occurrence of the string matching the value in the <b>Source</b> field. The matching string and replacing string can be different lengths.
	Source	Literal string to search for.
	Insert	Blank
Add an OWNER parameter	Target	Blank
	Source	Blank
	Insert	Value for the OWNER parameter.

Template Type	Field	Description
		<b>Note:</b> There must be no OWNER in the input BIND command, and the following control statement must be input to the plan lookup program CMNDB2PL: AUTHORITY=OWNER,INSERT
Add a QUALIFIER parameter	Target	Blank
	Source	Blank
	Insert	Value for the QUALIFIER parameter.
		<b>Note:</b> There must be no QUALIFIER in the input BIND command, and the following control statement must be input to the plan lookup program CMNDB2PL: INSERTQUAL

#### Note

The entries on this panel are not restricted by entries on the **Db2 Logical Subsystem nickname Bind Process Templates** at the global level. The entries at the global level are only a model for applications.

Define templates, then press **Enter** to accept panel entries.

Create a logical subsystem for every promotion level and production environment in the application where the Db2 Option will manage Db2 components.

7. On the **Db2 Logical Subsystems** panel, type **B** on the Line Command for a logical subsystem row and press **Enter**.

The **Db2 Logical Subsystem nickname Bind General Templates** panel CMNLD2L5 is displayed.

#### Note

This facility is not part of the Core Db2 option. Only named templates are available if you choose to use the Core option.

CMNLD2L5 Db2 Logical Subsystem QAD1 Bind General Templates Row 1 to 13 of 13  
 Command ==> Scroll ==> CSR

Token name	+ Target template	+ Source template	+
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	
_____	_____	_____	

8. On the **Db2 Logical Subsystems** panel, type **Q** on the Line Command for a logical subsystem row and press **Enter**. The **Db2 Logical Subsystem *nickname* SQL Templates (Named)** panel CMNGD2L3 is displayed:

CMNGD2L3 Db2 Logical Subsystem PROD SQL Process Templates (Named)  
 Command ==> \_\_\_\_\_

Templates	Target	Source	Deploy
Schema . . . .	PROD-rrrrrrrrrrrrrrrr	+ _____	+
Collection . .	_____	+ _____	+
WLM . . . . .	_____	+ _____	+
Location . . .		+ _____	+ _____
Qualifier . .	PROD-rrrrrrrrrrrrrrrr	+ _____	+ PROD
Owner . . . .	SERD-rrrrrrrrrrrrrrrr	+ _____	+ SERD

9. On the **Db2 Logical Subsystems** panel, type **G** on the Line Command for a logical subsystem row and press **Enter**. The **Db2 Logical Subsystem *nickname* SQL Process Templates (General)** panel is displayed:

CMNGD2L4 B2 Logical Subsys PROD SQL Process Templates (Ge Row 1 to 21 of 21)  
 Command ==> \_\_\_\_\_ Scroll ==> CSR

Token name	Target template	Source template	
_____	+ _____	+ _____	+
_____	+ _____	+ _____	+
_____	+ _____	+ _____	+
_____	+ _____	+ _____	+
...			
_____	+ _____	+ _____	+
_____	+ _____	+ _____	+
***** Bottom of data *****			

Refer to the explanation of the Global equivalent for these - [CMNGD2L4 B2 Logical Subsystem PROD SQL Process Templates \(Ge Row 1 to 21 of 21\)](#).

10. On the **Db2 Logical Subsystems** panel, type **V** on the Line Command for a logical subsystem row and press **Enter**. The **Db2 Logical Subsystem nickname Bind Service Named Templates** panel CMNGD2L6 is displayed:

```
CMNGD2L6 Db2 Logical Subsystem QAD1 Bind Service Named Templates
Command ==>

Templates      Target      Source

Collection . . _____ + _____ +
Qualifier . . . _____ + _____ +
Owner . . . . . _____ + _____ +
```

These three named parameters are templated in the same way as for package/plan binds (see section 6 above). The underlying processes are different.

11. On the **Db2 Logical Subsystems** panel, type **H** on the line command for a logical subsystem row and press **ENTER**. The **Db2 Logical Subsystem nickname BIND Service General Templates** panel CMNLD2L7 is displayed, from which the BIND Service process general tokens for that particular logical subsystem can be specified:

```
CMNLD2L7 Db2 Logical Subsystem QAD1 Bind Service General Temp Row 1 to 13 of 13
Command ==>                                     Scroll ==> CSR

      Token name      + Target template      + Source template      +
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
```

This general token processing is described in step 7 above. However, there are some extra features which are applicable only to Bind Service processing.

When a grant SQL component is associated with the bind service process and the token name is >GRANTEE<, the template is applied to the list of grantee users/groups on the grant SQL supplied to the process.

Also, if you enclose the source template in single quotes, the target template is only used if the value to be replaced exactly matches the source template value.

## Define Application Active Libraries

Automated Db2 Option functions are activated when libraries managed by ChangeMan ZMF are changed in promotion and production environments. These libraries are defined as active libraries in the Db2 Option. See [Active Libraries](#) for details.

### Note

For stored procedures, user defined functions, and triggers, use production libraries instead of baseline libraries for Db2 active libraries at installation.

Follow these steps to define active libraries in your Db2 application:

On the **application - Db2 Administration Options** panel, choose option **2 Library** and press **Enter** to display the **Db2 Active Library List** panel:

```

CMNLD2AL          Db2 Active Library List          Row 1 to 10 of 10
Command ==>> _____ Scroll ==>> CSR

      Logical      Bind
      name         /SQL   Db2 active library name
____ SERT6        B      CMNTP.S6.V810.PROD.DBB
____ SERT6        B      CMNTP.S6.V810.PROD.DBR
____ SERT6        B      CMNTP.S6.V810.PROD.LOD
____ SERT6        B      CMNTP.S6.V810.PROD.PKG
____ SERT6        S      CMNTP.S6.V810.PROD.SPD
____ SERT6        S      CMNTP.S6.V810.PROD.SPN
____ SERT6        S      CMNTP.S6.V810.PROD.SPQ
____ SERT6        S      CMNTP.S6.V810.PROD.STL
____ SERT6        S      CMNTP.S6.V810.PROD.TRG
____ SERT6        S      CMNTP.S6.V810.PROD.UDF
____ SERT6        V      /cmntp/S6/V810/Prod00/BSG
____ SERT6        V      /cmntp/S6/V810/Prod00/BSZ
***** Bottom of data *****

```

The following table describes fields on the **Db2 Active Library List** panel:

Field	Description
Line	Type a line command to the left of a panel row.
Command	
	I Insert a blank row.
	R Repeat an existing row.
	D Delete an existing row.
Logical name	Type a Db2 logical subsystem, or connector, name that is already defined in Db2 application administration.



Field	Description
	Type * to see the Db2 Logical Subsystem List panel from which you may select a logical subsystem name. The Logical Name is also called Db2 <i>nickname</i> in this manual.
Type	Type an indicator for the kind of processing invoked when the active library is updated:  <b>B</b> Activate Db2 bind processing.  <b>S</b> Activate stored procedure processing, which includes stored procedures, user defined functions, and triggers.  <b>V</b> Activate Db2 Bind Service processing.
Db2 active library name	Type the fully qualified data set name of a library or zFS directory name, that is monitored for change and will invoke Db2 Option bind, stored procedure, or bind service processing. Library types and subtypes that should be coded as active libraries:  Bind processing - BIND PLAN command library, BIND PACKAGE command library, Load module library, and DBRM libraries  Stored procedure processing - Stored procedure library. See the table below.  Bind service processing - BIND SERVICE command library GRANT authority to service DDL library

This table shows you how the BND/SQL/SERVICE field relates to library types and other Db2 Option parameters. BND/SQL/SERVICE values are shown in **bold**.

Db2 Component	Like	Target Type	Sel Opt	Sub Typ	BIND/SQL/Service
Db2 Application Program Source	S	Db2 Program Load			
Db2 Application Program Load	L				<b>**B</b>
DBRM	P		D	R	<b>**B</b>

<b>Db2 Component</b>	<b>Like</b>	<b>Target Type</b>	<b>Sel Opt</b>	<b>Sub Typ</b>	<b>BIND/ SQL/ Service</b>
BIND PLAN Command	P		D	B	**B
BIND PACKAGE Command	P		D	P	**B
External Stored Procedure Source	S	Stored Procedure Load			
External SQL Stored Procedure Source	S	Stored Procedure Load	D	Q	**S
External Stored Procedure Load	L		D	S	**B & S
Native SQL Stored Procedure	P		D	N	**S
General DDL (e.g. CREATE PROCEDURE for external SP)	P		D	D	**S
User Defined Function Definition	P		D	D	**S
Trigger Definition	P		D	T	**S
BIND Service command	P		D	V	**V
Service GRANT command	P		D	V	**V

\* Db2 Active Library specification for BIND plan/pkg (B), Process SQL (S), and Bind Service (V).

## Define Application Db2 Library Subtypes

Db2 library subtypes invoke special processing for Db2 components. When you defined the Application Library Types for these components, you coded D in the Selectable Option field. Here you assign a Db2 Sub Type to each of those library types.

On the **application - Db2 Administration Option** panel, choose option **3 Libtypes** and press **Enter**. The **application - Db2 Library Types** panel (CMNDLLT0) is displayed:

```

CMNDLLT0          STEV - Db2 Library Types          Row 1 to 12 of 12
Command ==>>> _____ Scroll ==>> CSR

Lib              Sub      End SQL
type  Description type      sentence
DBR   Db2 DBRM's      R         -
DBP   Db2 Bind Package Commands      P         -
PKG   Db2 Package Bind Control        P         -
DBB   Db2 Bind Plan Commands          B         -
STL   Db2 External Stored Procedure Load S         -
XPQ   Db2 External SQL stored proc Source D         #
SPD   Db2 Stored Procedure Definition D         #
NSQ   Native SQL Stored Procedures    N         #
DDL   Data Definition Language        D         @
BSP   Bind Service PDSE based components V         -
BSZ   Bind Service zFS based components V         -
BSG   Bind Service Grant components   V         -
***** Bottom of data *****

```

This table describes the fields on the **application - Db2 Library Types** panel:

Field	Description
Lib type	Displays the library types from the Global Library Type definition that are defined with a Selectable Option of D.
Description	Displays the definition from the global library type.
Sub type	Type the Db2 Sub Type for special Db2 Option processing. Sub type processing is described in the next table below.
	B BIND PLAN command
	D CREATE statements for stored procedures, external user defined functions.
	N Native SQL stored procedure
	P BIND PACKAGE command
	S External SQL stored procedure load modules, REXX stored procedures
	T Trigger definition source

Field	Description
	V BIND SERVICE command or associated GRANT DDL.
End SQL sentence	Type an alternate SQL statement terminator. If the components in this library type include SQL that uses the semicolon (;) as a statement terminator, specify an alternate terminator for the stored procedure or function so that the semicolon is passed through to the server.
	You can specify any character except the following:
	- comma
	- underscore
	- single quote
	- double quote
	- left hand parenthesis
	- right hand parenthesis
	If you leave this field blank, the default alternate SQL statement terminator is semicolon (;).

Define the Db2 sub types, then press **Enter** to accept panel entries.

This table shows the processing assigned by Db2 Option library sub types:

Sub Type	Description	Modified Process	Sub Type Processing Description
B	BIND PLAN command	Promote Demote Install Backout	Process with plan lookup program CMNDB2PL to template BIND parameters for the target Db2 subsystem.
D	CREATE statements for stored procedures, external user defined functions.	Promote Demote Install Backout	Process with utility program CMNDB2DD to register the object in the Db2 catalog. Issue a DROP before the CREATE.

<b>Sub Type</b>	<b>Description</b>	<b>Modified Process</b>	<b>Sub Type Processing Description</b>
N	Native SQL stored procedures	Promote Demote Install Backout	Process with utility program CMNDB2DD to register the stored procedure in the Db2 catalog.
P	BIND PACKAGE command	Promote Demote Install Backout	Process with plan lookup program CMNDB2PL to find applicable BIND PLAN members and template parameters for the target Db2 subsystem.
Q	SQL stored procedure source	Promote Demote Install Backout	Process with utility program CMNDB2DQ to remove SQL procedural code, then process the CREATE with utility program CMNDB2DD to register the stored procedure in the Db2 catalog.
R	DBRM		Process with plan lookup program CMNDB2PL to find applicable BIND PACKAGE and BIND PLAN members and template BIND parameters for the target Db2 subsystem.
S	Stored procedure load modules, REXX stored procedures	Promote Demote Install Backout	If the Recycle Stored Procedure field is YES, issue Db2 command VARY WLM...REFRESH in the WLM-managed address space to refresh the executable.
T	Trigger definition source	Promote Demote Install Backout	Process with utility program CMNDB2DD to extract the table/event/time combinations. If Maintain Trigger Sequence YES, query SYSIBM.SYSTRIGGERS with utility program CMNDB2TR to find multiple triggers defined for the same table/event/time combination, then drop and recreate those triggers to maintain the original firing order.

Sub Type	Description	Modified Process	Sub Type Processing Description
V	BIND SERVICE command and associated GRANT SQL	Promote Demote Install Backout	Process with CMNDB2SV to apply templates to incoming BIND SERVICE and GRANT components. The templated BIND SERVICE command is passed to an IKJEFT01 step for processing. Any templated GRANT DDL is passed to CMNDB2GR for action.

 **Note**

If you create a change package and then change an existing library type definition, the new library type behavior is not expressed in the package. If you create a change package and then define a new library type, the library type behavior in the package is set when the first component is checked out in that library type.

This table shows you how the Db2 Option sub types relate to library types and other Db2 Option parameters. Sub types are shown in **bold**.

Db2 Component	Like	Target Type	Sel Opt	Sub Typ	BIND/SQL
Db2 Application Program Source	S	Db2 Program Load			
Db2 Application Program Load	L				B
DBRM	P		D	<b>**R</b>	B
BIND PLAN Command	P		D	<b>**B</b>	B
BIND PACKAGE Command	P		D	<b>**P</b>	B
External Stored Procedure Source	S	Stored Procedure Load			
External SQL Stored Procedure Source	S	Stored Procedure Load	D	<b>**Q</b>	S

Db2 Component	Like	Target Type	Sel Opt	Sub Typ	BIND/SQL
External Stored Procedure Load	L		D	**S	B & S
Native SQL Stored Procedure	P		D	**N	S
General DDL (e.g. CREATE PROCEDURE for external SP)	P		D	**D	S
User Defined Function Definition	P		D	**D	S
Trigger Definition	P		D	**T	S
BIND Service command	P		D	**V	V
Service GRANT command	P		D	**V	V

\* Db2 Active Library specification for BIND plan/pkg (B), Process SQL (S), and Bind Service (V).

## Define Source/Target Connector

The BIND DEPLOY mechanism for distributing Native SQL stored procedures requires both a source and a target Db2 environment. This option is used to "connect" a source logical subsystem to a target logical subsystem.

It is not part of the "core" Db2 option functionality.

On the **application - Db2 Administration Option** panel, choose option **4 Connector** and press **Enter**. The **application - Db2 Logical Subsystem Connectors** panel (CMNLD2CL) is displayed:

```

CMNLD2CL      Logical Subsystem Connectors for Appl - STEV      Row 1 to 1 of 1
Command ==>>                                     Scroll ==>> CSR

      Connector      Source      Target
      name           name       name       Description
      ABC           PRODLCL1  PRODLCL2  TEST
***** Bottom of data *****

```

The connector definition must already exist at the global level. You can enter an asterisk in the line command to get a list of pre-defined connectors from which you can select.

Values from the source logical subsystem are used to identify the stored procedure that will be deployed. Values from the target logical subsystem are used to specify the name and related attributes of the stored procedure when it is deployed to the target.

If you wish to see which values will be used for your choice of source and target, select the row once any update has been saved.

To invoke the BIND DEPLOY mechanism based on a specific connector name, make the connector name the subsystem name associated with the relevant SQL active library.

This table describes the fields on the **application - Db2 Logical Subsystem Connectors** panel:

Field	Description
Connector Name	The 8 character name by which this connector definition will be referenced. It must be unique within connector and logical subsystem names.
Source Name	An existing logical subsystem name describing the source location of the required bind deploy command.
Target Name	An existing logical subsystem name describing the target location of the required bind deploy command.
Description	Free form text describing this connector definition.

## Define Secondary Bind Requirements

This option is used to allow the administrator to associate secondary bind logical subsystem names with those of primary bind logical subsystems.

The secondary names will be added to a secondary bind logical subsystem ISPF table (CMNDB22N) from which additional bind steps may be created each time binds are generated for the primary logical subsystem.

On the **application - Db2 Administration Option** panel, choose option **5 Secondary** and press **Enter**. The **application - Db2 Secondary Binding panel** (CMNLD2LB) is displayed:



```

CMNLD2LB          Db2 Secondary Binding          Row 1 to 5 of 5
Command ==>          Scroll ==> CSR

Enter END command to save changes or CANCEL to exit without saving changes.
Enter * in either name field for local DB2 logical subsystem list.

  Primary   Secondary   Promotion
Logical    Logical    Level Name   Active Description (Secondary)
Name       Name
PRODUCTN  QA             30   QA           Y   QA           (2NDARY BINDS)
PRODUCTN  SYSTEST       20   SYSTEST      Y   SYSTEM      (2NDARY BINDS)
PRODUCTN  UNIT          10   UNIT         Y   UNIT        (2NDARY BINDS)
QA        SYSTEST       20   SYSTEST      Y   SYSTEM      (2NDARY BINDS)
QA        UNIT          10   UNIT         Y   UNIT        (2NDARY BINDS)
***** Bottom of data *****

```

This table describes the fields on the **application - Db2 Secondary Binding** panel:

Field	Description
Primary Logical Name	A pre-existing logical subsystem name where the primary bind, for this definition, will take place. You can enter an asterisk in this field to get a list of existing logical subsystem names.
Secondary Logical Name	A pre-existing logical subsystem name where the secondary binds, for this definition, will take place. You can enter an asterisk in this field to get a list of existing logical subsystem names.
Promotion Level	This is the promotion level you wish to be associated with the secondary logical subsystem for this secondary bind. This will be used, via the relevant skeleton, to create an appropriate library concatenation for DBRMLIB (etc.). You can enter an asterisk in this field to get a list of allowed promotion levels for the remote site associated with the secondary logical subsystem.
Promotion Name	This is the nickname associated with this promotion level (display field only).
Active	This definition will only be acted on if this field is set to Y.

Field	Description
Description	The description associated with the secondary logical subsystem name (display field only).

## Specify Application Db2 General Parameters

Set options for processing Db2 components.

On the **application - Db2 Administration Options** panel, choose option **G** General and press **Enter** to display the **application - Db2 General Parameters** panel (CMNLDPM0):

```

CMNLDPM0          ACTP - Db2 General Parameters
Command ==>> _____

Enter "/" to select option
  _ Use package name in Db2 PC version
  _ Force package name in Db2 PC version

```

This table describes the options and values on the **application - Db2 General Parameters** panel.

Field	Description
Use Package Name in Db2 PC version	Select with / to determine whether the VERSION field on the <b>Db2 Physical Subsystems</b> panel (CMNSTG18) is initialized to the package ID in stage and recompile.
	When selected, initializes the VERSION field on the <b>Db2 Physical Subsystems</b> panel to the package ID.
	When not selected, initializes the VERSION field on the <b>Db2 Physical Subsystems</b> panel to blank.
Force Pkg Name in Db2 PC version	Select this to determine whether the Package ID <i>must</i> be used for the VERSION parameter for the Db2 precompiler in stage and recompile. This option must be blank if the "Use Package Name in Db2 PC version" field is blank.
	If selected, then display the VERSION field in browse mode on the <b>Db2 Physical Subsystems</b> panel (CMNSTG18) in stage and recompile so that the field cannot be changed.
	If not selected, then display the VERSION field on the <b>Db2 Physical Subsystem</b> panel in edit mode so it can be changed.

Type your choices for the general parameters, then press **Enter** to accept panel entries.

Press **PF3** to return to the application - **Db2 Options Administration** panel.

## Customize Skeletons for Db2

---

You must customize a skeleton to set the DBRM library type for the Db2 precompiler, and you may want to modify skeletons that execute plan lookup program CMNDB2PL.

### Set DBRM Library Type for Db2 Precompile

---

Modify skeleton CMN\$\$VAR to set the library type for DBRM created by the Db2 Precompiler in skeleton CMN\$\$PDB.

1. Make sure you are working in a CUSTOM SKELS library and not the delivered skeleton library.
2. Edit member CMN\$\$VAR.
3. Find "SET PREFERRED LIBTYPES".
4. Read the instructions under that heading and choose a method to set the library type for DBRM in one or more applications.
5. Uncomment the provided code, or code your own solution to set variable DBRMLTP.

### Modify Plan Lookup Parameters In Skeletons

---

Plan lookup program CMNDB2PL is included in seven skeletons. The function of program CMNDB2PL is controlled by keyword parameters input through the SYSIN ddname. The values of these keywords in the delivered skeletons will be suitable for many user sites. However, there is an interaction between these keyword options and the definition of logical subsystems in the Db2 Option.

See [CMNDB2PL - BIND Utility](#) for detailed descriptions of each CMNDB2PL keyword option. Modify the skeletons below if you need to change the values for these options:

<b>Skeleton</b>	<b>Purpose</b>
CMN\$ \$PRB	Bind Db2 plans and packages for promotion/ demotion at local sites.
CMN\$ \$PB2	Secondary binding at local sites.
CMN\$ \$RPB	Bind Db2 plans and packages for promotion/ demotion at remote sites.

<b>Skeleton</b>	<b>Purpose</b>
CMN\$ \$RB2	Secondary binding at remote sites.
CMN\$ \$SPB	Bind Db2 plans and packages for promotion/demotion using shadow promotion libraries at the development ChangeMan ZMF instance.
CMN\$ \$SB2	Secondary binding using shadow promotion libraries.
CMN21	Bind Db2 plans and packages for install at production ChangeMan ZMF instances.
CMN\$ \$IB2	Secondary binding for install in production.
CMN32	Bind Db2 plans and packages for baseline ripple at development ChangeMan ZMF instances.
CMN\$ \$BB2	Secondary binding for baseline ripple.
CMN49	Bind Db2 plans and packages for backout at production ChangeMan ZMF instances.
CMN56	Bind Db2 plans and packages for backout (reverse baseline ripple) at development ChangeMan ZMF instances.

## SQL Processing In Skeletons

Modify the skeletons below if you need to change the way SQL is processed - observe the comments in each. Refer also to [Stored Procedure Utilities](#):

<b>Skeleton</b>	<b>Purpose</b>
CMN\$ \$PSQ	Local promote/demote - This routine presents SQL/DDDL components to local Db2.
CMN\$ \$RSQ	Remote promote / demote - This routine presents SQL/DDDL components to remote Db2.

<b>Skeleton</b>	<b>Purpose</b>
CMN\$ \$SQL	Install/backout - This routine processes install and backout for SQL/DDL components to Db2.

## Bind Service Processing In Skeletons

Modify the skeletons below if you need to change the way SQL is processed - observe the comments in each. Refer also to [Bind Service Support](#) for more information.

<b>Skeleton</b>	<b>Purpose</b>
CMN\$ \$BSV	Install/backout - This routine templates BIND SERVICE command parameters as well as related GRANT DDL. The templated bind commands are processed by IKJEFT01 and the templated GRANT DDL is presented to Db2.
CMN\$ \$PSV	The same as CMN\$\$BSV but used for local promote/demote.
CMN\$ \$RSV	The same as CMN\$\$BSV but used or remote promote/demote.

## Installation in Other Db2 Subsystems

You must bind the DBRM for program CMNDB2SQ in all Db2 subsystems where ChangeMan ZMF will perform binds or manage stored procedures, user defined functions, and triggers for promote, demote, install, and backout.

A ZMF license is required on every LPAR that runs CMNDB2PL.

If you also intend to use ZMF support for native SQL versions and bind deploy then there are further binds required. These are detailed in the supplied DB2OPTN and DB2OPTNR sample JCL components.

# 4. DB2 Component Processing

---

This chapter describes how the Db2 Option processes application components.

- [Library Types and Sub Types](#)
- [CREATE versus ALTER](#)
- [Component Processing Summary](#)

## Library Types and Sub Types

---

Application Db2 component processing is defined by:

- Like processing for the library type
- Db2 Option library sub type

## Library Types

Like all other component processing in ChangeMan ZMF, basic processing for Db2 components at stage are determined by the *like* parameter of the library type definition. For example, like-source components are processed by a transform procedure to create executables.

Additional processing for Db2 components is determined by sub-types you assign to a library type in Db2 Option administration. (Prior to ChangeMan ZMF 8.1 some Db2 component processing was determined by reserved library types.)

Setting the Selectable Option to D on a library type definition makes that library type available for setting a sub-type in Db2 Option administration.

Db2 Component	Like	Target Type	Sel Opt	Sub Typ	BIND/SQL\
Db2 Application Program Source	S	Db2 Program Load			
Db2 Application Program Load	L				B
DBRM	P		D	R	B

Db2 Component	Like	Target Type	Sel Opt	Sub Typ	BIND/SQL\
BIND PLAN Command	P		D	B	B
BIND PACKAGE Command	P		D	P	B
External Stored Procedure Source	S	Stored Procedure Load			
External SQL Stored Procedure Source	S	Stored Procedure Load	D	Q	S
External Stored Procedure Load	L		D	S	B & S
Native SQL Stored Procedure	P		D	N	S
General DDL (e.g. CREATE PROCEDURE for external SP)	P		D	D	S
User Defined Function Definition	P		D	D	S
Trigger Definition	P		D	T	S

This table shows library types you must define to process various kinds of Db2 components.

\* Db2 Active Library specification for BIND/SQL action.

## Library Sub Types

<b>Sub Type</b>	<b>Description</b>	<b>Modified Process</b>	<b>Sub Type Processing Description</b>
B	BIND PLAN statements	Promote Demote Install Backout	Process with plan lookup program CMNDB2PL to template BIND parameters for the target Db2 subsystem. <b>Note:</b> Only 1 PLAN per member can be specified. Refer to Chapter 6, "CMNDB2PL - BIND Utility" as a reference to how CMNDB2PL adds Plans and Packages to the 'to be bound' list.
D	CREATE statements for stored procedures, external user defined functions.	Promote Demote Install Backout	Process with utility program CMNDB2DD to register the object in the Db2 catalog. Issue a DROP before the CREATE.
N	Native SQL stored procedure definitions	Promote Demote Install Backout	Drop/Create, Alter add version, and bind deploy mechanisms supported for this library type.
P	BIND PACKAGE statements	Promote Demote Install Backout	Process with plan lookup program CMNDB2PL to find applicable BIND PLAN members and template parameters for the target Db2 subsystem.
Q	SQL stored procedure source	Promote Demote Install Backout	Process with utility program CMNDB2DQ to remove SQL procedural code, then process the CREATE with utility program CMNDB2DD to register the stored procedure in the Db2 catalog.
R	DBRM		Process with plan lookup program CMNDB2PL to find applicable BIND PACKAGE and BIND PLAN members and template BIND parameters for the target Db2 subsystem.



Sub Type	Description	Modified Process	Sub Type Processing Description
S	Stored procedure load modules, REXX stored procedures	Promote Demote Install Backout	If the Recycle Stored Procedure field is YES, issue Db2 commands VARY WLM...REFRESH in the WLM-managed address space to refresh the executable.
T	Trigger definition source	Promote Demote Install Backout	Process with utility program CMNDB2DD to extract the table/event/time combinations. If Maintain Trigger Sequence YES, query SYSIBM.SYSTRIGGERS with utility program CMNDB2TR to find multiple triggers defined for the same table/event/time combination, then drop and recreate those triggers to maintain the original firing order.

Db2 Option library sub types enable additional processing for Db2 components at promote, demote, install, and backout. This table describes that processing by sub type.

## CREATE versus ALTER

There are several ways to change the Db2 catalog definition of a stored procedure, user defined function, or trigger.

1. DROP the Db2 object and execute a CREATE with the new definition.
2. Execute an ALTER that includes only the parameters you want to modify in the definition.
3. For Native SQL SPs you can use ALTER to ADD a new VERSION.

For all but Native SQL SPs, we recommend that you always stage a CREATE member to change a definition. The Db2 Option issues a DROP automatically before it executes a CREATE definition.

If you use CREATE, then the latest version of the complete object definition is available in baseline for checkout. If you use ALTER, the next time you check out the definition, you can only see the parameters in the definition that you changed in the last ALTER. You can look in prior versions kept by ChangeMan ZMF for the complete definition, but you will have these difficulties.

4. To come up with a complete, current definition, you have to apply the changes in each ALTER to the last CREATE definition.

5. If you install more ALTERs for an object than the number of prior versions of the component type kept by ChangeMan ZMF, the CREATE will be lost.

There are cases where you cannot use CREATE. When a Db2 object has dependencies, it can be difficult to DROP the object.

When dependencies exist, you may have no alternative than to use ALTER to change the definition. If you must use ALTER, we recommend that you include the entire CREATE definition as comments inside the ALTER SQL member.

For Native SQL SPs the same considerations apply but, in addition, you may wish to use ALTER ADD VERSION

To add a new version of an SP to the target Db2 catalog. The ZMF Db2 option supplies mechanisms to aid with the automation of activating new and dropping old versions of native SQL SPs.

## Component Processing Summary

---

This section provides a summary of the processing performed by the ChangeMan ZMF base product and the Db2 option for each kind of Db2 component.

## Programs with Imbedded SQL

---

### Library Types

Component	Lib Type or Like	Sub Type
Source	Like-S	
DBRM	Like-P	R
Load	Like-L	

### Stage (Like-S)

**Language** is a high-level language such as COBOL or assembler.

**Db2 Precompile** is YES.

Action	Object
Copy to staging library (if not there)	Source
Delete from staging library	DBRM Load

Action	Object
Precompile (extract SQL)	DBRM
Compile with high level language procedure.	Object
Link edit object	Load
Build relationship to source	PM records
Copy to staging library	DBRM Load

## Audit

Action	Object
Examine relationships to package components	
Examine relationships to baseline components	

## Promote/Demote

Action	Object
Copy members to promotion library	DBRM Load
Determine which plans/packages to bind	
Template BIND command	BIND
Execute bind	Db2 Catalog

## Install/Backout

Action	Object
Baseline ripple or copy to production library	Source DBRM Load
Determine which plans/packages to bind	
Template BIND command	BIND

Action	Object
Execute bind	Db2 Catalog

## BIND Commands

### Library Types

Component	Lib Type or Like	Sub Type
Plan BIND commands	Like-P	B
Package BIND commands	Like-P	P

### Stage (Like-P)

Action	Object
Copy to staging library (if not there)	

### Audit

Action	Object
Examine relationships to baseline components	

### Promote/Demote

Action	Object
Copy members to promotion library	
Template BIND command	BIND
Execute bind	Db2 Catalog

### Install/Backout

Action	Object
Baseline ripple or copy to production library	
Template BIND command	BIND

Action	Object
Execute bind	Db2 Catalog

## Procedure Definition DDL

---

### Library Types

Component	Lib Type or Like	Sub Type
Procedure Definition	Like-P	D

### Stage (Like-P)

Action	Object
Copy to staging library (if not there)	Procedure Defn

### Audit

Action	Object
Examine relationships to baseline components	

### Promote/Demote

Action	Object
Copy members to promotion library	Procedure Defn
Implement Procedure Definition	Db2 Catalog

### Install/Backout

Action	Object
Baseline ripple or copy to production library	Procedure Defn

Action	Object
Implement Procedure Definition	Db2 Catalog

## Stored Procedures – External

### Library Types

Component	Lib Type or Like	Sub Type
Source	Like-S	
DBRM	Like-P	R
Load	Like-L	S

### Stage (Like-S)

**Language** is a high-level language such as COBOL or assembler.

**Db2 Precompile** is YES.

Action	Object
Copy to staging library (if not there)	Source
Delete from staging library	DBRM Load
Precompile (extract SQL)	DBRM
Compile with high level language procedure	Object
Link edit object	Load
Build relationship to source	PM records
Copy to staging library	DBRM Load

### Audit

Action	Object
Examine relationships to package components	

Action	Object
Examine relationships to baseline components	

### Promote/Demote

Action	Object
Copy members to promotion library	DBRM Load
Determine which plans/packages to bind	
Template BIND command	BIND
Execute bind	Db2 Catalog
VARY WLM,APPLENV= <i>envname</i> ,REFRESH	WLM-managed address space

### Install/Backout

Action	Object
Baseline ripple or copy to production library	Source DBRM Load
Determine which plans/packages to bind	
Template BIND command	BIND
Execute bind	Db2 Catalog
VARY WLM,APPLENV= <i>envname</i> ,REFRESH	WLM-managed address space

## Stored Procedures – External SQL

### Library Types

Component	Lib Type or Like	Sub Type
Source	Like-S	Q
DBRM	Like-P	R

Component	Lib Type or Like	Sub Type
Load	Like-L	S

## Stage (Like-S)

Language invokes procedure **CMNSQL**.

Db2 Precompile is **YES**.

Action	Object
Copy to staging library (if not there)	Source
Delete from staging library	DBRM Load
Convert to SQL source to C	C-source
Precompile C (extract SQL)	DBRM
Compile with C language procedure	Object
Link edit object	Load
Build relationship to source	PM records
Copy to staging library	DBRM Load

## Audit

Action	Object
Examine relationships to package components	
Examine relationships to baseline components	

## Promote/Demote

Action	Object
Copy members to promotion library	Source DBRM Load
Determine which plans/packages to bind	
Template BIND command	BIND
Execute bind	Db2 Catalog
Extract Procedure Definition DDL from source SQL	Procedure Defn
Implement Procedure Definition	Db2 Catalog



Action	Object
VARY WLM,APPLENV=envname,REFRESH	WLM-managed address space

## Install/Backout

Action	Object
Baseline ripple or copy to production library	Source DBRM Load
Determine which plans/packages to bind	
Template BIND command	BIND
Execute bind	Db2 Catalog
Extract Procedure Definition DDL from source SQL	Procedure Defn
Implement Procedure Definition	Db2 Catalog
VARY WLM,APPLENV=envname,REFRESH	WLM-managed address space

## Stored Procedures - Native SQL

Native SQL stored procedure deployment mechanisms:

- Drop/Create
- Alter Add Version
- Bind Package Deploy

There is no build process for these components. All processing takes place at the Db2 subsystem which is the target for the promotion/install/etc. For BIND DEPLOY the relevant bind command is issued to the target Db2 subsystem but routed for execution to the source subsystem. The BIND DEPLOYS from the source to the target.

## Library Types

Component	Lib Type or Like	Sub Type
Source	Like-P	N

## Stage

Action	Object
Copy to staging library (if not there)	Source

## Audit

Action	Object
No Action	

## Promote/Demote

Action	Object
Copy member to promotion library	Source
Implement procedure definition (via templates if defined)	Source, where bind deploy not in use
Deploy procedure definition	Bind deploy from source Db2 catalog
Activate procedure version	If Db2 versioning in use

## Install/Backout

Action	Object
Copy member to production library	Source
Implement procedure definition (via templates if defined)	Source, where bind deploy not in use
Deploy procedure definition	Bind deploy from source Db2 catalog

Action	Object
Activate procedure version	If Db2 versioning in use

## Stored Procedures – REXX

REXX stored procedures:

- Require REXX support to be active in the target Db2 subsystem. (The DBA or system programmer configures this support.)
- Run only in WLM-managed address spaces.

During the build process, all you have to do is place the source for the REXX stored procedure in the SYSEXEC concatenation of the started task associated with the target WLM-managed address space.

### Library Types

Component	Lib Type or Like	Sub Type
Source	Like-P	S

### Stage

Action	Object
Copy to staging library (if not there)	Source

### Audit

Action	Object
Examine relationships to baseline components	

### Promote/Demote

Action	Object
Copy members to promotion library	Source

Action	Object
VARY WLM,APPLENV=envname,REFRESH	WLM-managed address space

## Install/Backout

Action	Object
Baseline ripple or copy to production library	Source
VARY WLM,APPLENV=envname,REFRESH	WLM-managed address space

## User Defined Functions - External

### Library Types

Component	Lib Type or Like	Sub Type
Source	Like-S	
DBRM	Like-P	R
Load	Like-L	

### Stage (Like-S)

Language is high level language such as COBOL or assembler.

Db2 Precompile is YES.

Action	Object
Copy to staging library (if not there)	Source
Delete from staging library	DBRM Load
Precompile (extract SQL)	DBRM
Compile	Object
Link edit object	Load
Build relationship to source	PM records

Action	Object
Copy to staging library	DBRM Load

## Audit

Action	Object
Examine relationships to package components	
Examine relationships to baseline components	

## Promote/Demote

Action	Object
Copy members to promotion library	DBRM Load
Determine which plans/packages to bind	
Template BIND command	BIND
Execute bind	Db2 Catalog

## Install/Backout

Action	Object
Baseline ripple or copy to production library	Source DBRM Load
Determine which plans/packages to bind	
Template BIND command	BIND
Execute bind	Db2 Catalog

## User Defined Functions - Source

---

## Library Types

Component	Lib Type or Like	Sub Type
Function Definition	Like-P	D

## Stage (Like-P)

Action	Object
Copy to staging library (if not there)	Function Defn

## Audit

Action	Object
Examine relationships to baseline components	

## Promote/Demote

Action	Object
Copy members to promotion library	Function Defn
Implement Function Definition	Db2 Catalog

## Install/Backout

Action	Object
Baseline ripple or copy to production library	Function Defn
Implement Function Definition	Db2 Catalog

## Database Triggers

---

### Library Types

Component	Lib Type or Like	Sub Type
Trigger Definition	Like-P	T
Stage (Like-P)		
Action		Object

Component	Lib Type or Like	Sub Type
Copy to staging library (if not there)	Trigger Defn	

### Audit

Action	Object
Examine relationships to baseline components	

### Promote/Demote

Action	Object
Copy members to promotion library	Trigger Defn
Implement Trigger Definition	Db2 Catalog
Reorder Triggers in correct sequence	Db2 Catalog

### Install/Backout

Action	Object
Baseline ripple or copy to production library	Trigger Defn
Implement Trigger Definition	Db2 Catalog
Reorder Triggers in correct sequence	Db2 Catalog

# 5. Native SQL SP Lifecycle

---

This chapter discusses various actions in the lifecycle.

- Checkin/Stage
- Promote
- Demote
- Install
- Backout
- Skeleton changes (overview)

## Checkin/Stage

---

### Stage From Development

---

There is a new option on the 'Stage from Development' panel CMNSTG02:

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

      Package: STEV000288      Status: DEV      Install date: 20161120
      Change rqst: 00000001      Location: HERE

ISPF Library:
Project . . . . . ZMFSD
Group . . . . . DB2
Type . . . . . JCL
Member . . . . . _____ (Blank/pattern for list; \* for all members)

Other partitioned, sequential or HFS dataset:
DSN . . . . . _____ +
Org . . . . . _____ (PDS, Seq, PAN, LIB, Oth, HFS)

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

Enter "/" to select option
 / Confirm request           _ Expand HFS subdirectories
 _ Lock component           _ Display component user options
 / Extract Stored Procedure from Db2 catalog
```

If you choose that option, you are presented with a new panel from which you can stage the SP into the package:



```

CMNSTG25                               Stage Native-SQL SP from Db2
Command ==> _____

Package: STEV000288           Status: DEV           Install date: 20161120

Stored Procedure:
  Db2 id . . . . DSN
  Location . . . _____
  Schema . . . . ZMFSD _____ +
  Name . . . . NTVSQL01 _____ +
  Version . . . _____ +
  Version Ind . _____

Component:
  Name . . . . NTVSQL01
  Library type. NSQ

Enter "/" to select option
  / Add package information to component
  _ Lock component in package

```

The fields available on this panel (CMNSTG25) are:

Stored Procedure	Description
Db2 id	Enter the subsystem id of the Db2 instance you would like ZMF to contact to look for the stored procedure (SP) to be staged into the package. If location is left blank then the SP must reside in the catalog of this Db2 instance. This Db2 subsystem must be contactable by ZMF and must be at least Db2 v11.
Location	If the Stored Procedure is located at a remote Db2 instance you can specify its location here. The usual requirements for DRDA access to remote Db2 tables, from the Db2 ID specified above, must be in place.
Schema	This is the (up to 128 byte) schema name used to identify the SP definition to be extracted.
Name	This is the (up to 128 byte) name of the SP.
Version	This is the (up to 64,122 for DBCS, byte) Db2 version identifier for the Native SQL SP you wish to extract. If this, and the next, field is left blank then ZMF will extract the version with the greatest routineid.
Version In	This indicator is used only if the Version field is left blank. It allows the user to extract one of the First, Latest, or Active versions of the SP. You can enter the whole word or just the first character i.e. F, L, or A. any other value is invalid:

Stored Procedure	Description
	<b>First:</b> The version of the SP with the earliest CREATEDTS will be staged.
	<b>Latest:</b> The version of the SP with the latest of either CREATEDTS or ALTEREDTS will be staged.
	<b>Active:</b> The currently active version of the SP will be staged.

Component	Description
Name	The 8 byte name of the component to be staged into the package. This is usually the same name as the SP itself (and leaving this field blank will ensure this is so). If the SP name is longer than 8 bytes then you will have to choose an 8 byte component name yourself.
Library Type	The library type to be used to stage the component. This must be a Db2 indicated library type with a subtype of N.

Option	Description
Add package information to component	If selected this will result in comments being added to the top of the extracted SP component. These comments identify the package being used as the vehicle for the deployment of this SP ("The first three (optional) comment lines have been added by ZMF.").
Lock component in package	Selecting this option will cause the component to be locked in the package. This sets the same variable as the same option on CMNSTG02.

If the SP component is being entered directly via ISPF edit then the member can be staged directly into the package.

In order to enable client-pack, batch, and general access to this, this new function has been implemented as an extension to the existing CHECKIN service.

All these values may, optionally, be controlled via new HLL exits taken at new points for this new function. The internal exit names are:

**BULD0025 pre-panel CMNSTG25**

**BULD0125 post-panel CMNSTG25**

The data interface for the checkin/build function (BULD) has been extended to pass these new attributes (more info in the next section).

## Promote

---

Promotion of the SP results in its deployment to the target Db2 subsystem. The target of the promotion must be covered by a logical Db2 subsystem definition and, by default, the standard templating provided by that definition will be applied.

There are 3 methods which may be used to promote (and install) native SQL SPs:

## Drop/Create

---

This process does not differ to a great extent from the existing promotion mechanism for DDL in general. Facilities exist whereby the administrator can provide their own validation and manipulation routine. Changes required for test promotion levels (and, quite often, different target production environments) are made by the promotion (or installation) mechanism as the component is applied to the target environment. This is the methodology that the existing Db2 templating mechanisms follow. The facilities for native SQL SPs extend this method to allow the administrator to take control of both validation and manipulation of the SP component. This is implemented as an HLL exit point from CMNDB2DD where control is passed to a routine of the sites choosing (specified via SYSIN parameters). The incoming DDL is already allocated to the step and is available to be read from that ddname. The exit can request the termination of the process (and issue error messages), it will also be in a position to replace the incoming DDL values and pass the updated DDL either directly to Db2 (i.e. let CMNDB2DD apply the DDL directly) or output it to a dataset to be passed on to whichever Db2 processing utility the user site wishes to use. This is not a 'standard' HLL exit because the promotion job must be able to run on an LPAR other than the one where HLLX is running, and there may be no direct means of communicating back to ZMF in order to drive HLLX, so the exit is called directly from the execution of CMNDB2DD. The invocation method is very similar to standard HLLX, you will be able to code this exit in REXX or any LE-language.

## Alter Add Version

---

If versioning has been requested in the admin definition for the logical subsystem then a row indicating the current active version will be written to the 'Db2 Object Attribute Table' to allow for a potential future demote action. A separate (optionally held) job will be submitted to activate the new version of all Native SQL SPs included in this promotion. If allowed to execute immediately then the changed SP will be activated at time of promote, else at a time later when the job is released. Indication of the current active version (i.e. prior to activating the just promoted version) is kept in the local CMNZMF.CMNDB2\_ATTRIBS table.

These actions are implemented jointly between CMNDB2DD and CMNDB2AV. The keyword SPVERSION=YES is used to have CMNDB2DD generate activation transactions which are written to an external file. CMNDB2AV picks up these transactions and, based on their content, proceeds to save prior active version information and then activate the newly promoted SP.

## Bind Deploy

---

When this deployment method is chosen we will route the relevant bind deploy command to the originating Db2 subsystem as indicated in the admin definitions. This is achieved from CMNDB2DD running on the target site and issuing a remote call to IBMs stored procedure called ADMIN\_COMMAND\_DSN. The location of the source subsystem will be provided on the call to this SP and it will issue the bind deploy command to the source Db2. The usual remote promotion job which executes at the target site will contain the ALTER request required to activate the new version of the SP.

These actions are, again, implemented by a combination of CMNDB2DD and CMNDB2AV. CMNDB2DD issues the BIND DEPLOY command (which is routed via a call to srcllocation.SYSPROC.ADMIN\_COMMAND\_DSN) as a result of the SPVERSION=BOTH keyword. This results in version activation transactions being written to an external file and the BIND DEPLOY command being issued. Note that CMNDB2DD waits for confirmation that the

BIND DEPLOY has worked before proceeding. If it fails for some reason then messages are written to the CMNDB2DD SYSPRINT dataset and it finishes with rc=8. As a result, the CMNDB2AV activation process does not run. If all goes well then the BIND DEPLOY will have distributed the new version of the SP to the target Db2 subsystem and the ensuing CMNDB2AV execution will activate it.

## Demote

---

### Drop/Create

---

The current action for a demote of a DDL/SQL component is that CMNDB2DD will attempt to re-instate the prior version of the component via DROP/CREATE. To do this it searches the concatenation allocated to the SQLIN ddname (which is built starting from the next highest level library), using the first matching member name it finds to do the create. If the member name was not found when searching the SQLIN concatenation then CMNDB2DD picks up the DDL from the STGLIB ddname (which points at the current level promotion library) and uses the information found within to issue a DROP request only (i.e. it assumes that this SP was a new one and there is nothing to replace it with once it has been demoted).

Once CMNDB2DD has completed the target component is then removed from the current level promotion library.

Some sites may prefer to use mechanisms other than CMNDB2DD to process the DDL. In order to make this easier an option is available to pass the DDL built by CMNDB2DD to an external file which can then be processed by whatever utility the site wishes (this was also mentioned above for promote).

## Alter Add Version

---

If versioning is active then CMNDB2DD will not attempt to drop/create using the concatenation hierarchy. Instead we will use the information maintained in the 'Db2 Object Attribute' table to take steps to activate the prior version. The version being demoted will be dropped and the SP component will be removed from the promotion library. The relevant row in the 'Db2 Object Attribute' table will also be deleted.

CMNDB2DD keyword SPVERSION=YES is used to generate the relevant activation transactions required for CMNDB2AV to activate the recorded prior version for this SP. The version being demoted is also dropped.

## Bind Deploy

---

SPVERSION=UNDO causes CMNDB2DD to generate version activation transactions which will reverse the activation sequence so that the prior recorded version is activated. This is different from SPVERSION=YES only in that different templates are used to identify the schema of the target SP as there are two values to take into account (the source and target Db2 locations).

## Install

---

The install processes specific to Native SQL SPs are similar to those for promotion. The differences being in the (usual) way that install jobs are distributed and scheduled.

Install usually implements a 'promotion cleanup' for executable components. This is based on the concatenated execution environment approach, i.e. once the component has been installed into production then you don't need separate copies in all the test environments. This is, generally, not appropriate for SPs and we need to leave the promotion/test Db2 alone. Note that ChangeMan ZMF allows for this to be controlled via promotion admin definitions (see details below).

## Install skeleton changes

---

The install mechanism for Stored Procedures (and other Db2 objects such as UDFs and Triggers) has been divorced from CMN20 and implemented as part of CMN21.

Some advantages of this are:

### 1) Job routing:

Job cards specific to the target Db2 subsystem can be defined for Db2 Bind processing. These are different to the job cards for rest of Promote/Install jobs. Note that LPAR on which the target Db2 runs may be different from the one where the rest of the installation runs.

Refer to panel CMNGD2S1, reached via =A.G.O.2 then 1 - Identify Db2 Physical Subsystems, then select the row desired to update or create the Jobcards.

```
CMNGD2S1          Db2 Physical Subsystems - Part 2 of 2
Command ==> _____

Db2 subsystem:  DSN
Site:          U810DP
Load Library:  DSNB10.SDSNLOAD

Job statement information for Db2 binds:
//DB2FORDP JOB (X170,374),GLOBAL,CLASS=A,MSGCLASS=X,
//          REGION=0M,TIME=(10)
//*_____
/*JOBPARM S=S0W1
```

See also skeleton CMN\$SD2J

## 2) Db2 skeleton variables:

The full range of Db2 variables which may be used in skeletons are only generated if there is a need to generate a CMN21 job (i.e. if there are binds to be done). For a package containing just DDL related components none of these variables are available.

## Backout

---

Backout is similar to demote with the obvious difference in the way the jobs are structured.

Once the SP backout is successful then the active version row for this subsystem and this package will be removed (it will be re-established should the SP be installed again).

If versioning is not active then backout will consist of DROP/CREATE using the re-instatedfrom-backup version of the SP component.

### Backout skeleton changes

The backout mechanism for Stored Procedures (and other Db2 objects such as UDFs and Triggers) has been divorced from CMN50 and implemented as part of CMN49. The same advantages apply as for the install skeletons.

## Skeleton changes (overview)

---

There are 3 main functional skeletons related to Native SQL stored procedures. These are the ones already in use for general Db2 object management, viz:

```
CMN$$PSQ Local promote/demote
CMN$$RSQ Remote promote/demote
CMN$$SQL Install
```

These have been changed to add new sections dedicated to the management of Native SQL SPs. Existing facilities have been left in place and are selected via `\&DB2SUBT NE N`, the new sections require `\&DB2SUBT EQ N`.

Additional sysin parameters are specified for the CMNDB2DD step which relate specifically to Native SQL SP management. Some of these are driven by the admin settings for the target logical subsystem and some are simply hardcoded in the skeleton.

If SP version management is required (driven via admin and the ISPF variable `\&SQACTV EQ YES`) then a job is submitted from this process which will activate the required version of the SP. The job makes use of the CMN\$\$D2J jobcard skeleton, which is filled out with information related to the physical Db2 subsystem hosting the target logical subsystem (from Global Db2 option admin). A facility is in place whereby, if not already present, a TYPRUN=HOLD parameter will be added to the jobcard (given sufficient room on the last active jobcard line). This is done by setting ISPF variable `\&D2TYPR` to a value of HOLD prior to imbedding CMN\$\$D2J, this is how the skeleton is delivered.

Use of the CMN\$\$SQL skeleton has been moved from CMN20 (general install) and CMN50 (general backout) to CMN21 (Db2 specific install) and CMN49 (Db2 specific backout).

# 6. Templating Examples

---

This chapter provides examples of how you can achieve your BIND command and SQL templating needs, using the fields on the **Db2 Logical Subsystem BIND or SQL Templates** panel in the ChangeMan ZMF Db2 Option. Refer also to [Examples](#) for a detailed account of setting up for native SQL stored procedure versioning and the bind deploy mechanisms.

- [Templated BIND Command Parameters](#)
- [Templated DDL/SQL](#)
- [Templating Examples](#)
- [BIND PLAN Examples](#)
- [BIND PACKAGE Example](#)
- [General token templates](#)

## Templated BIND Command Parameters

---

A BIND command member is obtained from the staging libraries in your change package. If the BIND command member is not staged, it is obtained from baseline libraries. (Promotion libraries are searched after staging libraries if the BIND is performed for promotion or demotion).

Templates defined in the logical subsystem in the Db2 Option are applied to provide BIND parameters that are suitable for the target Db2 environment.

You can alter these parameters in a BIND command by using templating:

- PLAN Name
- PACKAGE Location
- PACKAGE Name/Collection ID
- OWNER
- QUALIFIER



### Important

BIND Command Keyword Option Order The ChangeMan ZMF Db2 Option uses IBM service routine IKJPARS to parse BIND commands. This ensures that Db2 Option processing is synchronized with changes that IBM might make to BIND keyword operands.

IKJPARS does not attempt to maintain the order of keyword operands in a BIND command that it parses. Therefore, keyword operands sent to IKJEFT01 from the ZMF Db2 Option may be in a different order than in the original BIND command member in a staging, promotion, baseline, or production library.

## Templated DDL/SQL

---

DDL/SQL to define or change a Db2 stored procedure, user defined function, or trigger is obtained from the staging libraries in your change package. Unlike BIND command members, DDL/SQL is never actioned from promotion or baseline libraries.

Templates defined in the logical subsystem in the Db2 Option are applied to provide DDL/ SQL that is suitable for the target Db2 environment.

You can modify many parameters in the definitions for stored procedures, triggers, and user defined functions by using both named parameter templates and general token templates (described elsewhere in this manual). The CMNDB2DD HLL exit is also available for DDL/SQL validation and manipulation.

## Templating Examples

---

Typical templating examples include:

- Replacing characters at an offset
- Adding characters at the end
- Replacing characters at the end
- Deleting characters at the end
- Replacing a character string with another string
- Adding an OWNER parameter
- Adding a QUALIFIER parameter

## Replace Characters At an Offset

Use ? (question marks) in the Target field to define the offset of characters you want to change in a parameter value.

Input Parameter Value	Logical Subsystem Fields		Templated Output Value
	Target	Source	
	**Insert		
ABCDEFGG	??X??**YZ		ABXDE**YZ
ABCDEFGG	**X		XBCDEFG

## Add Characters at the End

Use ? (question marks) in the Target field to define an offset that is as long or longer than an input parameter value, followed by characters that you want to add to the end of the value.

Input Parameter Value	Logical Subsystem Fields		Templated Output Value
	Target	Source	
	**Insert		
ABCDEFGG	???????**XYZ		ABCDEFGG**XYZ
ABCDEFGG	????????**X		ABCDEFGG**X

## Replace Characters at the End

Use \* (asterisk) followed by n characters in the Target field to define the n characters you want to replace at the end of a parameter value.

Input Parameter Value	Logical Subsystem Fields		Templated Output Value
	Target	Source	
	**Insert		
ABCDEFGG	**XYZ		ABCD**XYZ

Input Parameter Value	Logical Subsystem Fields	Templated Output Value
ABCDEFGFG	***X	ABCDEF**X

## Delete Characters at the End

Use ~ (not) in the Target field to specify a parameter value character that you want to replace with a space. Since spaces are not valid in the middle of a parameter value, use ~ to delete characters at the end of a value.

Input Parameter Value	Logical Subsystem Fields		Templated Output Value
	Target	Source	
	**Insert		
ABCDEFGFG	????**~---		ABCD
ABCDEFGFG	**WXYZ~-----		**WXYZ

## Replace a Character String with Another String

Use the Source field to specify a string to be replaced in a parameter value and use the Target field to define the string to replace it. The search string and the replace string may be different lengths.

Input Parameter Value	Logical Subsystem Fields		Templated Output Value
	Target	Source	
	**Insert		
ABCDEFGFG	XYZ	**ABC	XYZDEFG
ABCDEFGFG	WXYZ	**DEF	ABCWXYZG

## Add an Owner Parameter

Specify a value in the Insert field to add an OWNER parameter and value. To insert an OWNER parameter, the following must be true:

There is no OWNER= parameter in the input BIND command.

This control statement is input to the plan lookup program CMNDB2PL at DDname CMNPLCTL:

```
AUTHORITY=OWNER,INSERT
```

Input Parameter Value	Logical Subsystem Fields		Templated Output Value
	Target	Source	
	**Insert		
		**XYZ	**OWNER(XYZ)

## Add a Qualifier Parameter

Specify a value in the Insert field to add a QUALIFIER parameter and value. To insert a QUALIFIER parameter, the following must be true:

There is no QUALIFIER= parameter in the input BIND command.

- This control statement is input to the plan lookup program CMNDB2PL at DDname CMNPLCTL:

*BIND PLAN Example*

INSERTQUAL

Input Parameter Value	Logical Subsystem Fields		Templated Output Value
	Target	Source	
	**Insert		
		**XYZ	**QUALIFIER(XYZ)

## BIND PLAN Example

This section presents a simple example of how a production BIND PLAN command can be modified by templates in the Db2 Option so that the same application can be bound for unit testing, systems testing, and production execution in the same Db2 subsystem.

[CMNDB2PL - BIND Command](#) describes the process used by plan lookup program CMNDB2PL to determine what plans and packages need to be bound and to locate the required BIND command member. This example assumes that the BIND PLAN command member is staged in the change package that is being promoted and installed.

This is the production BIND PLAN command in member PRDAPPL1, which is staged in the change package:

```

BIND PLAN(PRDAPPL1)      -
PKLIST (COLLP.\*)       -
QUALIFIER(APPL1P)       -
ACTION (REPLACE)        -
ISOLATION (CS)          -
RETAIN                   -
EXPLAIN (NO)             -
VALIDATE(BIND)           -
ACQUIRE(USE)            -
RELEASE (COMMIT)

```

These are the active libraries defined for the application in the Db2 Option on panel CMNLD2AL:

```

CMNLD2AL                      Db2 Active Library List                      Row 1 to 9 of 9
Command ===\> _____ Scroll ===\> CSR

   Logical   Bind
   name      /SQL      Db2 active library name
____ UNIT    B          CMNTP.UNIT.ACTP.LOADLIB
____ UNIT    B          CMNTP.UNIT.ACTP.PKGBIND
____ UNIT    B          CMNTP.UNIT.ACTP.PLANBIND
____ SYST    B          CMNTP.SYST.ACTP.LOADLIB
____ SYST    B          CMNTP.SYST.ACTP.PKGBIND
____ SYST    B          CMNTP.SYST.ACTP.PLANBIND
____ PROD    B          CMNTP.PROD.ACTP.LOADLIB
____ PROD    B          CMNTP.PROD.ACTP.PKGBIND
____ PROD    B          CMNTP.PROD.ACTP.PLANBIND

***** Bottom of data *****

```

## Promote to Unit Test

When the change package containing BIND PLAN member PRDAPPL1 is promoted to the unit test level, PRDAPPL1 in the staging library is copied to library CMNTP.UNIT.ACTP.PLANBIND.

### Note

For a new output BIND OWNER command when OWNER is not present, requires the CMN\$ \$PRM skeleton to be modified with the INSERT value for the AUTHORITY= statement. See [Add an Owner Parameter](#).

This is an active library for the application in the Db2 Option. This active library is associated with the UNIT logical subsystem, so the BIND command in member PRDAPPL1 is templated according to the rules in logical subsystem UNIT.

This is the **Db2 Logical Subsystem UNIT Templates** panel CMNGD2L2:

```

CMNGD2L2                               Db2 Logical Subsystem UNIT Templates
Command ===\> _____

Templates          Target                Source                Insert
General:
Schema . . . . . _____ + _____ + _____
Qualifier . . . . ????T                + _____ + _____ +
Bind owner . . . . _____ + _____ + UNIT                +
WLM Env . . . . . _____ + _____ + _____


Plan:
Name . . . . . TST                        PRD

Package:
Location . . . . _____ + _____ + _____
Collection . . . ????T                + _____ + _____

```

Staged BIND PLAN command member PRDAPPL1 is compared to the BIND command after CMNDB2PL applies templates from logical subsystem UNIT:

Input BIND Command	Output BIND Command
BIND PLAN(PRDAPPL1) -	BIND PLAN(TSTAPPL1) -
PKLIST (COLLP.*) -	PKLIST (COLLT.*) -
QUALIFIER(APPL1P) -	QUALIFIER(APPL1T) -
ACTION (REPLACE) -	ACTION (REPLACE) -
ISOLATION (CS) -	ISOLATION (CS) -
RETAIN -	RETAIN -
EXPLAIN (NO) -	EXPLAIN (NO) -
VALIDATE(BIND) -	VALIDATE(BIND) -
ACQUIRE(USE) -	ACQUIRE(USE) -
RELEASE(COMMIT)	RELEASE(COMMIT) -
	OWNER(UNIT)

 **Note**

The order of keyword options in the output BIND command may not match the input order. See [BIND Command Keyword Option Order](#).

The templated BIND command is executed in the promotion job.

## Promote to System Test

When the change package containing BINDPLAN member PRDAPPL1 is promoted to the system test level, PRDAPPL1 in the staging library is copied to library CMNTP.SYST.ACTP.PLANBIND.

This is an active library for the application in the Db2 Option. This active library is associated with the SYST logical subsystem, so the BIND command in member PRDAPPL1 is templated according to the rules in logical subsystem SYST.

This is the **Db2 Logical Subsystem SYST Templates** panel:

CMNGD2L2		Db2 Logical Subsystem SYST Templates	
Command ===\> _____			
Templates	Target	Source	Insert
General:			
Schema . . . . .	_____	+ _____	+ _____
Qualifier . . . .	?????S	+ _____	+ _____ +
Bind owner . . . .	_____	+ _____	+ SYST +
WLM Env . . . . .	_____	+ _____	+ _____
Plan:			
Name . . . . .	SYS	PRD	
Package:			
Location . . . . .	_____	+ _____	+ _____
Collection . . . .	????S	+ _____	+ _____

Staged BIND PLAN command member PRDAPPL1 is compared to the BIND command after CMNDB2PL applies templates from logical subsystem SYST:

Input BIND Command	Output BIND Command
BIND PLAN(PRDAPPL1) -	BIND PLAN(SYSAPPL1) -
PKLIST (COLLP.*) -	PKLIST (COLLS.*) -
QUALIFIER(APPL1P) -	QUALIFIER(APPL1S) -
ACTION (REPLACE) -	ACTION (REPLACE) -
ISOLATION (CS) -	ISOLATION (CS) -
RETAIN -	RETAIN -
EXPLAIN (NO) -	EXPLAIN (NO) -
VALIDATE(BIND) -	VALIDATE(BIND) -
ACQUIRE(USE) -	ACQUIRE(USE) -
RELEASE(COMMIT)	RELEASE(COMMIT) -

<b>Input BIND Command</b>	<b>Output BIND Command</b>
	OWNER(SYST)

**Note**

The order of keyword options in the output BIND command may not match the input order. See [BIND Command Keyword Option Order](#).

The templated BIND command is executed in the promotion job.

## Install and Baseline Ripple

When the change package containing BIND PLAN member PRDAPPL1 is baselined, PRDAPPL1 in the staging library is copied to library CMNTP.PROD.ACTP.PLANBIND.

This is an active library for the application in the Db2 Option. This active library is associated with the PROD logical subsystem, so the BIND command in member PRDAPPL1 is templated according to the rules in logical subsystem PROD.

This is the **Db2 Logical Subsystem PROD Templates** panel:

```

CMNGD2L2                Db2 Logical Subsystem PROD Templates
Command ==>\> _____

Templates      Target          Source          Insert
General:
Schema . . . . . _____ + _____ + _____
Qualifier . . . _____ + _____ + _____ +
Bind owner . . . _____ + _____ + PROD +
WLM Env . . . . . _____ + _____ + _____

Plan:
Name . . . . . _____          _____

Package:
Location . . . _____ + _____ + _____
Collection . . _____ + _____ + _____

```

Staged BIND PLAN command member PRDAPPL1 is compared to the BIND command after CMNDB2PL applies templates from logical subsystem PROD:

<b>Input BIND Command</b>	<b>Output BIND Command</b>
BIND PLAN(PRDAPPL1) -	BIND PLAN(PRDAPPL1) -
PKLIST (COLLP:*) -	PKLIST (COLLP:*) -



Input BIND Command	Output BIND Command
QUALIFIER(APPL1P) -	QUALIFIER(APPL1P) -
ACTION (REPLACE) -	ACTION (REPLACE) -
ISOLATION (CS) -	ISOLATION (CS) -
RETAIN -	RETAIN -
EXPLAIN (NO) -	EXPLAIN (NO) -
VALIDATE(BIND) -	VALIDATE(BIND) -
ACQUIRE(USE) -	ACQUIRE(USE) -
RELEASE(COMMIT)	RELEASE(COMMIT) -
	OWNER(PROD)

 **Note**

The order of keyword options in the output BIND command may not match the input order. See [BIND Command Keyword Option Order](#).

The templated BIND command is executed in the installation job.

## BIND PACKAGE Example

This section presents a simple example of how a production BIND PACKAGE command can be modified by templates in the Db2 Option so that the same DBRM can be bound for unit testing, systems testing, and production execution in the same Db2 subsystem.

[CMNDB2PL - BIND Command](#) describes the process used by plan lookup program CMNDB2PL to determine what plans and packages need to be bound and to locate the required BIND command member. This example assumes that the BIND PACKAGE command member is in the baseline library and that no other binds are required.

This is the production BIND PACKAGE command stored in the baseline library for application program PROGRAM1:

```

BIND PACKAGE (COLLP)          -
MEMBER (PROGRAM1)             -
QUALIFIER (APPL1P)           -
SQLERROR (NOPACKAGE)         -
VALIDATE (BIND)               -
ISOLATION (CS)                -
RELEASE (DEALLOCATE)         -
EXPLAIN (YES)                 -
FLAG (I)                      -
ENABLE (*)                     -
ACTION (REPLACE)

```

These are the active libraries defined for the application in the Db2 Option:

```

CMNLD2AL                      Db2 Active Library List                      Row 1 to 9 of 9
Command ===\> _____ Scroll ===\> CSR

   Logical   Bind
   name      /SQL      Db2 active library name
____ UNIT    B         CMNTP.UNIT.ACTP.LOADLIB
____ UNIT    B         CMNTP.UNIT.ACTP.PKGBIND
____ UNIT    B         CMNTP.UNIT.ACTP.PLANBIND
____ SYST    B         CMNTP.SYST.ACTP.LOADLIB
____ SYST    B         CMNTP.SYST.ACTP.PKGBIND
____ SYST    B         CMNTP.SYST.ACTP.PLANBIND
____ PROD    B         CMNTP.PROD.ACTP.LOADLIB
____ PROD    B         CMNTP.PROD.ACTP.PKGBIND
____ PROD    B         CMNTP.PROD.ACTP.PLANBIND

***** Bottom of data *****

```

## Promote to Unit Test

When the change package containing PROGRAM1 is promoted to the unit test level, the load module for Db2 program PROGRAM1 is copied to library CMNTP.UNIT.ACTP.LOADLIB.

This is an active library for the application in the Db2 Option. This active library is associated with the UNIT logical subsystem, so the BIND PACKAGE command in baseline member PROGRAM1 is templated according to the rules in logical subsystem UNIT.

This is the **Db2 Logical Subsystem UNIT Templates** panel CMNGD2L2:

```

CMNGD2L2                               Db2 Logical Subsystem UNIT Templates
Command ===\> _____

Templates          Target                Source                Insert
General:
Schema . . . . . _____ + _____ + _____
Qualifier . . . . ????T                + _____ + _____ +
Bind owner . . . . _____ + _____ + UNIT          +
WLM Env . . . . . _____ + _____ + _____

Plan:
Name . . . . . TST                      PRD

Package:
Location . . . . _____ + _____ + _____
Collection . . . ????T                + _____ + _____

```

Baseline BIND PACKAGE command member PROGRAM1 is compared to the BIND PACKAGE command after CMNDB2PL applies templates from logical subsystem UNIT.

Input BIND Command	Output BIND Command
BIND PACKAGE(COLLP) -	BIND PACKAGE(COLLT) -
MEMBER(PROGRAM1) -	MEMBER(PROGRAM1) -
QUALIFIER(APPL1P) -	QUALIFIER(APPL1T) -
SQLERROR(NOPACKAGE) -	SQLERROR(NOPACKAGE) -
VALIDATE(BIND) -	VALIDATE(BIND) -
ISOLATION(CS) -	ISOLATION(CS) -
RELEASE(DEALLOCATE) -	RELEASE(DEALLOCATE) -
EXPLAIN(YES) -	EXPLAIN(YES) -
FLAG(I) -	FLAG(I) -
ENABLE(*) -	ENABLE(*) -
ACTION(REPLACE)	ACTION(REPLACE) OWNER(UNIT)

 **Note**

The order of keyword options in the output BIND command may not match the input order. See [BIND Command Keyword Option Order](#).

The templated BIND PACKAGE command is executed in the promotion job.

## Promote to System Test

When the package is promoted to the system test level, the load module for Db2 program PROGRAM1 is copied to library CMNTP.TEST.ACTP.LOADLIB.

Since this is an active library in the Db2 Option, the BIND command for PROGRAM1 is templated according to the rules in the logical subsystem named SYST.

This is the **Db2 Logical Subsystem SYST Templates** panel CMNGD2L2:

```

CMNGD2L2                Db2 Logical Subsystem SYST Templates
Command ===\> _____

Templates                Target                Source                Insert
General:
Schema . . . . . _____ + _____ + _____
Qualifier . . . . ????S                + _____ + _____ +
Bind owner . . . . _____ + _____ + SYST                +

WLM Env . . . . . _____ + _____ + _____

Plan:
Name . . . . . SYS                PRD

Package:
Location . . . . _____ + _____ + _____
Collection . . . . ????S                + _____ + _____
  
```

Baseline BIND PACKAGE command member PROGRAM1 is compared to the BIND PACKAGE command after CMNDB2PL applies templates from logical subsystem SYST.

Input BIND Command	Output BIND Command
BIND PACKAGE(COLLP) -	BIND PACKAGE(COLLS) -
MEMBER(PROGRAM1) -	MEMBER(PROGRAM1) -
QUALIFIER(APPL1P) -	QUALIFIER(APPL1S) -
SQLERROR(NOPACKAGE) -	SQLERROR(NOPACKAGE) -
VALIDATE(BIND) -	VALIDATE(BIND) -
ISOLATION(CS) -	ISOLATION(CS) -
RELEASE(DEALLOCATE) -	RELEASE(DEALLOCATE) -
EXPLAIN(YES) -	EXPLAIN(YES) -
FLAG(I) -	FLAG(I) -
ENABLE(*) -	ENABLE(*) -
ACTION(REPLACE)	ACTION(REPLACE) -
	OWNER(SYST)

 **Note**

The order of keyword options in the output BIND command may not match the input order. See [BIND Command Keyword Option Order](#).

The templated BIND PACKAGE command is executed in the promotion job.

## Install and Baseline Ripple

When the package is baseline rippled, the load module for Db2 program PROGRAM1 is copied to library CMNTP.PROD.ACTP.LOADLIB.

Since this is an active library in the Db2 Option, the BIND command for PROGRAM1 is templated according to the rules in the logical subsystem named PROD.

This is the **Db2 Logical Subsystem PROD Templates** panel:

```
CMNGD2L2                               Db2 Logical Subsystem PROD Templates
Command ===\> _____


Templates          Target              Source              Insert
General:
Schema . . . . . _____ + _____ + _____
Qualifier . . . . _____ + _____ + _____ +
Bind owner . . . . _____ + _____ + PROD +
WLM Env . . . . . _____ + _____ + _____

Plan:
Name . . . . . _____

Package:
Location . . . . _____ + _____ + _____
Collection . . . _____ + _____ + _____
```

Baseline BIND PACKAGE command member PROGRAM1 is compared to the BIND PACKAGE command after CMNDB2PL applies templates from logical subsystem PROD.

Input BIND Command	Output BIND Command
BIND PACKAGE(COLLP) -	BIND PACKAGE(COLLP) -
MEMBER(PROGRAM1) -	MEMBER(PROGRAM1) -
QUALIFIER(APPL1P) -	QUALIFIER(APPL1P) -
SQLERROR(NOPACKAGE) -	SQLERROR(NOPACKAGE) -
VALIDATE(BIND) -	VALIDATE(BIND) -
ISOLATION(CS) -	ISOLATION(CS) -
RELEASE(DEALLOCATE) -	RELEASE(DEALLOCATE) -
EXPLAIN(YES) -	EXPLAIN(YES) -
FLAG(I) -	FLAG(I) -
ENABLE(*) -	ENABLE(*) -
ACTION(REPLACE)	ACTION(REPLACE) -
	OWNER(PROD)

 **Note**

The order of keyword options in the output BIND command may not match the input order. See [BIND Command Keyword Option Order](#).

The templated BIND PACKAGE command is executed in the installation job.

## General token templates

General token templates are provided to give the ability to implement your own automated changes for SQL and BIND components as they are moved through the lifecycle, without having to wait for specific keyword support to be programmed into the relevant utilities.

### SQL general token templating

Sysin supplied to CMNDB2DD to implement a general token template:

```
TOKENNAME=
TOKENSRCT=
TOKENTGTT=
```

The parameters allow one to define one's own SQL parameters to be templated via 'standard' CMNDB2DD templating. Freeform token processing will take place in addition to and after all the existing fixed name clause processing (e.g. after the likes of owner, qualifier etc. templating). Lists of subparameters are supported by applying the relevant template to each of the subparameters in turn. Subparameters in a list can be removed by having the source template match the individual list entry and using target template of >REMOVE\<.

TOKENNAME= specifies a string which will be looked for in the SQL. This may include imbedded blanks as long as the whole string is enclosed in single quotes. If there are no imbedded blanks then quotes are optional. Strings including imbedded blanks must not contain more than 5 subwords (and each subword must be 16 bytes or less – this should cover all sensible requirements).

In the SQL, to be recognized the token name may be preceded by either a blank or a comma, and followed by a blank or a left hand bracket.

The next word following the token will be templated according to the standard rules with the (optional) source template being supplied via TOKENSRCT and the (required) target template by TOKENTGTT.

As many of these groups as one needs may be specified. They are processed sequentially. The code applies the template to each found occurrence of TOKENNAME (it doesn't stop looking after the first found, only stopping when the current SQL sentence is exhausted). The resulting SQL is then subject to the next set of TOKEN templates and so on.

The TOKENNAME value may be up to 64 bytes The TOKEN template fields may be up to 128 bytes and can be specified across lines as per the other fixed name templates.

An example is:

```
TOKENNAME='ORDER BY'  
TOKENSRCT=NAME  
TOKENTGTT=CREATOR
```

This will look for the clause 'order by' in the SQL it will then look beyond that clause for the first word following on from there and, in this case, if it finds NAME it will replace it with CREATOR.

## BIND general token templating

As CMNDB2PL parses the BIND command as a TSO command, each command parameter, and its subparameters, are addressed as discrete variables in the program. As such, the general token templating feature implemented for the BIND process is different to that for the SQL process. Instead of parsing the command string as a whole looking for one or more parameter strings, we match the **exact** general token name against those known to the program.

Apart from this difference, BIND processing general templating is similar to that implemented for the SQL process.

Bind parameters currently supported by general token templating are given here:

```

ACQUIRE
CACHESIZE
COPY
CURRENTDATA
CURRENTSERVER
DBPROTOCOL
DEGREE
DISCONNECT
DYNAMICRULES
ENCODING
EXPLAIN
FLAG
IMMEDWRITE
ISOLATION
KEEPDYNAMIC
LIBRARY
MEMBER
OPTHINT
OWNER
PACKAGE name
PATH
PKLIST
PLAN name
QUALIFIER
RELEASE
REOPT
REPLVER
ROUNDING
SQLERROR
SQLRULES
VALIDATE

```

General token templates allow collection names in the PKLIST bind parameter to be individually templated and, as required, removed from the PKLIST.

This section presents 4 test examples, one plan and 3 package bind control components.

Plan member (DBB)

TEST1:

```

BIND PLAN(TEST1)          -
    PKLIST(CA_TNG_SUBROUTINES.*, -
            AD_SHR_ROUTINES.*, -
            CR_UTILITIES.*) -
    OWNER      (DBPPMGS)    -
    QUALIFIER  (DB2PMGS)    -
    VALIDATE   (RUN)        -
    ISOLATION  (CS)         -
    RELEASE    (COMMIT)     -
    EXPLAIN    (YES)        -
    ACTION     (REPLACE)    -
    PATH(DB2PRTB,MGS)
Package members (PKG)

```

TEST2:



```

BIND PACKAGE(TEST2CCA0P05) MEMBER(EI58VUSP) VALIDATE(BIND) -
  ISOLATION(CS) EXPLAIN(YES) CURRENTDATA(YES) -
  CURRENTSERVER(DB2DSNX) DEGREE(0) ACTION(REPLACE) -
  PATH(EPICPM05,EPICPMCR,EPICPMCN,EPICPMCF,EPICPMCS,EPICPXXX,EPICPYYY, -
    EPICPMCY,EPICPMCT,EPICPMCU,EPICPMCY,EPICPMCT,EPICPMCU)

```

#### TEST3:

```

BIND PACKAGE(TEST3) -
  OWNER (ASCMG) -
  MEMBER (EI58VUSP) -
  QUALIFIER (EPICP005) -
  VALIDATE (BIND) -
  ISOLATION (CS) -
  EXPLAIN (NO) -
  CURRENTDATA (YES) -
  DEGREE (1) -
  ACTION (REPLACE) -
  PATH
  (EPICPM05,EPICPMCR,EPICPMCN,EPICPMCF,EPICPMCS,EPICPXXX,EPICPYYY, -
    EPICPMCY,EPICPMCT,EPICPMCU, -
    EPICPMCY,EPICPMCT,EPICPMCU) |

```

#### TEST4:

```

BIND PACKAGE(TEST4) -
  OWNER(ASCMG) -
  MEMBER(EI58VUSP) -
  QUALIFIER(EPICP005) -
  VALIDATE(BIND) -
  ISOLATION(CS) -
  EXPLAIN(YES) -
  CURRENTDATA(YES) -
  CURRENTSERVER(XXXPROD) -
  DEGREE(1) -
  ACTION(REPLACE) -
  PATH(EPICPM05)

```

Here are the admin panels for a promotion logical subsystem for which there are active bind libraries defined. First the named templates:

```

CMNGD2L2          Db2 Logical Subsystem UNIT2 Bind Named Templates
Command ===> _____

Templates          Target          Source          Insert
General:
Schema . . . . . _____ + _____ + _____
Qualifier . . . . DEV             + _____ + DEV             +
Bind owner . . . . DEV            + _____ + DEV             +

Plan:
Name . . . . . _____

Package:
Location . . . . _____ + _____ + _____
Collection . . . _____ + _____ + _____

```

And the general templates:

```

CMNLD2AL          Db2 Logical Subsystem UNIT1 BIND General Template  Row 1 to 6 of 6
Command ===> _____ Scroll ===> CSR

Token name          + Target template          + Source template          +
_____ CURRENTSERVER    ???DSNT                    _____
_____ DEGREE            1                          _____
_____ EXPLAIN          YES                        _____
_____ PATH              ???U???                    _____
_____ PKLIST            >REMOVE<                   CR_UTIL
_____ PKLIST            UA_SHR                     AD_SHR

***** Bottom of data *****

```

This set of definitions created a CMNDB2PL step with CMNPLCTL input which looked like this:

```

TYPE=PROMOTE
  AUTHORITY=OWNER, INSERT
  INSERTQUAL
  *EARLYCHECK
  *IGNORENOSUBSYS
  *TRACE
  USEREXIT=(ASM, NOUNLOAD)
  USERID=SDOWNES
  PACKAGE=STEV000365
  PROJECT=STEV
  NOBASEDBBRC=12
  WARNINGRC=4
  USEDDB2PACKAGE
  *NOB2PLAN
  *FREEPLAN
  *CREATECC
  *IGNORENOBDRM
  *PKLTEMPLATE
  DB2ID=D20L
  LOGICAL=UNIT1
  PLANTGT=
  PLANSRC=
  PKGETGT=
  PKGESRC=
  LOCNTGT=
  LOCNSRC=
  QUALIFIER=DEV
  QUALTGT=DEV
  QUALSRC=
  OWNER=DEV
  OWNRTGT=DEV
  OWNRSRC=
  TOKENNAME=PKLIST
  TOKENSRCT=AD_SHR
  TOKENTGTT=UA_SHR
  TOKENNAME=PKLIST
  TOKENSRCT=CR_UTIL
  TOKENTGTT=
  >REMOVE<
  TOKENNAME=PATH
  TOKENSRCT=
  TOKENTGTT=???U???
  TOKENNAME=EXPLAIN
  TOKENSRCT=
  TOKENTGTT=YES
  TOKENNAME=DEGREE
  TOKENSRCT=
  TOKENTGTT=1
  TOKENNAME=CURRENTSERVER
  TOKENSRCT=
  TOKENTGTT=???DSNT
  REMOTEID=STEVEPRM

```

When the job ran the bind control output generated looked like this:

```

DSN SYSTEM(D20L )
  BIND PACKAGE(TEST2CCA0P05) +
  OWNER(DEV) +
  QUALIFIER(DEV) +
  PATH(+
    EPICUM05,+
    EPICUMCR,+
    EPICUMCN,+
    EPICUMCF,+
    EPICUMCS,+
    EPICUXXX,+
    EPICUYYY,+
    EPICUMCY,+
    EPICUMCT,+
    EPICUMCU,+
    EPICUMCY,+
    EPICUMCT,+
    EPICUMCU) +
  ACTION(REPLACE) +
  CURRENTDATA(YES) +
  CURRENTSERVER(DB2DSNT) +
  DEGREE(1) +
  EXPLAIN(YES) +
  ISOLATION(CS) +
  VALIDATE(BIND) +
  MEMBER(EI58VUSP)

```

```

BIND PACKAGE(TEST3) +
  OWNER(DEVMG) +
  QUALIFIER(DEVCP005) +
  PATH(+
    EPICUM05,+
    EPICUMCR,+
    EPICUMCN,+
    EPICUMCF,+
    EPICUMCS,+
    EPICUXXX,+
    EPICUYYY) +
  ACTION(REPLACE) +
  CURRENTDATA(YES) +
  DEGREE(1) +
  EXPLAIN(YES) +
  ISOLATION(CS) +
  VALIDATE(BIND) +
  MEMBER(EI58VUSP)

```

```

BIND PACKAGE(TEST4) +
  OWNER(DEVMG) +
  QUALIFIER(DEVCP005) +
  PATH(+
    EPICUM05) +
  ACTION(REPLACE) +
  CURRENTDATA(YES) +
  CURRENTSERVER(XXXDSNT) +
  DEGREE(1) +
  EXPLAIN(YES) +
  ISOLATION(CS) +
  VALIDATE(BIND) +
  MEMBER(EI58VUSP)

```

```

BIND PLAN(TEST1) +

```

```
OWNER (DEVPMGS) +
QUALIFIER (DEVPMGS) +
PATH (+
    DB2PUTB, +
    MGSU) +
RELEASE (COMMIT) +
ACTION (REPLACE) +
EXPLAIN (YES) +
ISOLATION (CS) +
VALIDATE (RUN) +
PKLIST (CA_TNG_SUBROUTINES.* , +
        UA_SHR_ROUTINES.\*)
END
```

## Notes

In the TEST2 package the owner and qualifier parameters were missing but have been inserted (as directed) by the templating process. In all other cases the owner and qualifier have had their first three characters overlaid by DEV. This is not new functionality, just a test of existing processes.

In all cases the fifth character of the path name has been replaced by U and this has occurred for each of the subparameters in the list. In TEST1 one of the PATH subparameters is not long enough to have its fifth character replaced by U (i.e. MGS) so the U has been appended (i.e. MGSU).

Where CURRENTSERVER is present it has had positions 4-7 replaced with DSNT

DEGREE has been set to 1

EXPLAIN has been set to YES.

The PKLIST in TEST1 has had the CR\_UTILITIES.\* entry removed (as it matched with one of the source templates for PKLIST where the target template was >REMOVE<). And the AD\_SHR\_ROUTINES.\* entry has been changed to UA\_SHR\_ROUTINES.\*

A production install logical subsystem was set up with the following admin settings, first the named templates:

```

CMNLD2AL      Db2 Logical Subsystem PRODN BIND Named Templates
Command ==> _____

  Templates      Target          Source          Insert
General:
Qualifier . . . . PROD          + _____ + PROD          +
Owner . . . . . PROD          + _____ + PROD          +

Plan:
Name . . . . . _____

Package:
Location . . . . _____ + _____ + _____
Collection . . . _____ + _____ + _____

```

And the general templates:

```

CMNLD2AL      Db2 Logical Subsystem UNIT1 BIND General Template  Row 1 to 6 of 6
Command ==> _____ Scroll ==> CSR

  Token name      + Target template      + Source template      +
_____ CURRENTSERVER  ???DSNP                _____
_____ DEGREE          0                      _____
_____ EXPLAIN        NO-                    _____
_____ PATH           >REMOVE<              EPICPXXX
_____ PATH           >REMOVE<              EPICPYYY
_____ PATH           ???P???
_____ PKLIST         CA_PRD                 CA_TNG

***** Bottom of data *****

```

When the same 4 components were installed the following bind control was generated:

```

DSN SYSTEM(D20L )
BIND PACKAGE(TEST2CCA0P05) +
OWNER(PROD) +
QUALIFIER(PROD) +
PATH(+
    EPICPM05,+
    EPICPMCR,+
    EPICPMCN,+
    EPICPMCF,+
    EPICPMCS,+
    EPICPMCY,+
    EPICPMCT,+
    EPICPMCU,+
    EPICPMCY,+
    EPICPMCT,+
    EPICPMCU) +
ACTION(REPLACE) +
CURRENTDATA(YES) +
CURRENTSERVER(DB2DSNP) +
DEGREE(0) +
EXPLAIN(NO) +
ISOLATION(CS) +
VALIDATE(BIND) +
MEMBER(EI58VUSP)
BIND PACKAGE(TEST3) +
OWNER(PRODG) +
QUALIFIER(PRODP005) +
PATH(+
    EPICPM05,+
    EPICPMCR,+
    EPICPMCN,+
    EPICPMCF,+
    EPICPMCS) +
ACTION(REPLACE) +
CURRENTDATA(YES) +
DEGREE(0) +
EXPLAIN(NO) +
ISOLATION(CS) +
VALIDATE(BIND) +
MEMBER(EI58VUSP)
BIND PACKAGE(TEST4) +
OWNER(PRODG) +
QUALIFIER(PRODP005) +
PATH(+
    EPICPM05) +
ACTION(REPLACE) +
CURRENTDATA(YES) +
CURRENTSERVER(XXXDSNP) +
DEGREE(0) +
EXPLAIN(NO) +
ISOLATION(CS) +
VALIDATE(BIND) +
MEMBER(EI58VUSP)
BIND PLAN(TEST1) +
OWNER(PRODMGS) +
QUALIFIER(PRODMGS) +
PATH(+
    DB2PPTB,+
    MGSP) +
RELEASE(COMMIT) +

```

```
ACTION(REPLACE) +
EXPLAIN(NO) +
ISOLATION(CS) +
VALIDATE(RUN) +
PKLIST(CA_PRD_SUBROUTINES.* , +
AD_SHR_ROUTINES.* , +
CR_UTILITIES.* )
END
```

## Notes

In the TEST2 package the owner and qualifier parameters were missing but have been inserted (as directed) by the templating process. In all other cases the owner and qualifier have had their first four characters overlaid by PROD. This is not new functionality, just a test of existing processes.

In all cases the fifth character of the path name has been replaced by P and this has occurred for each of the subparameters in the list. In TEST1 one of the PATH subparameters is not long enough to have its fifth character replaced by P (i.e. MGS) so the U has been appended (i.e. MGSP). The original bind component already had P in the 5th position in most cases.

Where CURRENTSERVER is present it has had positions 4-7 replaced with DSNP.

DEGREE has been set to 0.

EXPLAIN has been set to NO.

Values EPICPXXX and EPICPYYY have been removed where they have been found in any PATH list.

The PKLIST in TEST1 has had the CA\_TNG\_ROUTINES. entry *changed to* CA\_PRD\_ROUTINES.

Simple bindcntl can also be used for testing etc e.g.:



CBLDB201.PKG:

```
BIND PACKAGE(PROD) MEMBER(CBLDB201) ACT(REP) ISO(CS) -
      EXPLAIN(YES) VALIDATE(BIND) RELEASE(COMMIT) -
      QUALIFIER(PROD)
```

CBLDB201.DBB:

```
BIND PLAN(CBLDB201) -
      PKLIST(PROD.CBLDB201) -
      ACT(REP) -
      EXPLAIN(YES) -
      ISOLATION(CS) -
      QUALIFIER(PROD)
```

### Named templates:

CMNLD2L Db2 Logical Subsystem UNIT2 BIND Named Templates

Command ==>

Templates	Target	Source	Insert
General:			
Qualifier . . .	UNIT-???	+ _____	+ UNIT +
Owner . . . . .	SERD-???	+ _____	+ SERD +
Plan:			
Name . . . . .	_____		
Package:			
Location . . .	_____	+ _____	+ _____
Collection . .	UNIT	+ PROD	+ _____

### General templates:

CMNLD2L4 Db2 Logical Subsystem UNIT2 BIND General Template Row 1 to 4 of 4

Command ==> \_\_\_\_\_ Scroll ==> CSR

Token name	+ Target template	+ Source template	+
_____ EXPLAIN	NO~	_____	
_____ ISOLATION	UR	CS	
_____ PKLIST	UNIT	PROD	
_____ VALIDATE	RUN~	_____	

\*\*\*\*\* Bottom of data \*\*\*\*\*

### Bind output:

```

IKJ56644I NO VALID TSO USERID, DEFAULT USER ATTRIBUTES USED
READY
DSN SYSTEM(D20L )
DSN
  BIND PACKAGE(UNIT) OWNER(SERD) QUALIFIER(UNIT) RELEASE(COMMIT) ACTION(REP)
  EXPLAIN(NO) ISOLATION(UR) VALIDATE(RUN) MEMBER(CBLDB201)
DSNT254I -D20L DSNTBCM2 BIND OPTIONS FOR
  PACKAGE = D20L.UNIT.CBLDB201.( )
  ACTION          ADD
  OWNER           SERD
  QUALIFIER       UNIT
  VALIDATE        RUN
  EXPLAIN         NO
  ISOLATION       UR
  RELEASE         COMMIT
  COPY
  APREUSE
  APCOMPARE
  APRETAINDUP
  BUSTIMESENSITIVE YES
  SYSTIMESENSITIVE YES
  ARCHIVESENSITIVE YES
  APPLCOMPAT V12R1M500
  DESCSTAT YES
  APREUSESOURCE
DSNT255I -D20L DSNTBCM2 BIND OPTIONS FOR
  PACKAGE = D20L.UNIT.CBLDB201.( )
  SQLERROR        NOPACKAGE
  CURRENTDATA     NO
  DEGREE          1
  DYNAMICRULES
  DEFER
  NOREOPT         VARS
  KEEPDYNAMIC     NO
  IMMEDIATEWRITE INHERITFROMPLAN
  DBPROTOCOL      DRDA
  OPTHINT
  ENCODING        EBCDIC(00037)
  PLANMGMT        OFF
  PLANMGMTSCOPE  STATIC
  CONCURRENTACCESSRESOLUTION
  EXTENDEDINDICATOR
  PATH
DSNT275I -D20L DSNTBCM2 BIND OPTIONS FOR
  PACKAGE = D20L.UNIT.CBLDB201.( )
  QUERYACCELERATION
  GETACCELARCHIVE
  CONCENTRATESTMT
DSNT232I -D20L SUCCESSFUL BIND FOR
  PACKAGE = D20L.UNIT.CBLDB201.( )
DSN
  BIND PLAN(CBLDB201) OWNER(SERD) QUALIFIER(UNIT) ACTION(REP) EXPLAIN(NO)
  ISOLATION(UR) PKLIST(UNIT.CBLDB201)
DSNT252I -D20L DSNTBCM1 BIND OPTIONS FOR PLAN CBLDB201
  ACTION REPLACE
  OWNER SERD
  VALIDATE RUN
  ISOLATION UR
  ACQUIRE USE
  RELEASE COMMIT

```

```
EXPLAIN NO
DYNAMICRULES RUN
PROGAUTH DISABLE
DSNT253I -D20L DSNTBCM1 BIND OPTIONS FOR PLAN CBLDB201
NODEFER PREPARE
CACHESIZE 3072
QUALIFIER UNIT
CURRENTSERVER
CURRENTDATA NO
DEGREE 1
SQLRULES DB2
DISCONNECT EXPLICIT
NOREOPT VARS
KEEPDYNAMIC NO
IMMEDWRITE NO
DBPROTOCOL DRDA
OPTHINT
ENCODING EBCDIC(00037)
CONCURRENTACCESSRESOLUTION
PATH
DSNT200I -D20L BIND FOR PLAN CBLDB201 SUCCESSFUL
DSN
END
```

# 7. CMNDB2PL - BIND Utility

---

This chapter describes how program CMNDB2PL works and how to change its behavior by changing control statements that are input to the program.

- [Introduction](#)
- [CMNDB2PL DD Statements](#)
- [CMNDB2PL Operation](#)
- [Keyword Control Statements](#)
- [How CMNDB2PL Relates to ChangeMan ZMF](#)
- [CMNDB2PL Return Codes and Messages](#)
- [Sample CMNDB2PL Report](#)
- [Secondary Binding](#)

## Introduction

---

Program CMNDB2PL is the Plan Lookup program. This program is central to the ChangeMan ZMF Db2 Option facility that performs binds at promote, demote, install, and backout.

Plan Lookup program CMNDB2PL is designed to ensure a consistent Db2 plan/package environment as changes to Db2 application components progress through the ChangeMan ZMF package lifecycle to production.

Program CMNDB2PL queries the Db2 catalog tables looking for relationships between staged DBRMs and Db2 plans/packages in the catalog. These relationships indicate that a bind may be required.

CMNDB2PL also implements the ChangeMan Db2 logical subsystem concept, applying templates to staged or baselined BIND commands to derive the plan names and/or collection IDs which are relevant to the target logical subsystem.

The end product of this processing is a set of BIND command statements that is passed to a subsequent TSO step for execution in the target physical Db2 subsystem.

## CMNDB2PL DD Statements

---

The table in this section describes DD statements used by program CMNDB2PL to collect information and output BIND commands.

Variable *stssys* is embedded in some ddnames for CMNDB2PL, and it is resolved in ISPF file tailoring for the promote, demote, install, and backout skeletons. *stssys* is the Db2 physical subsystem where the binds are executed.

DDNAME	Description
CMNPLCTL	Input for CMNDB2PL control statements. See <a href="#">Keyword Control Statements</a> for control statement formats and definitions.
CMNPLDBB	Input for a list of BIND PLAN members in the change package. Member statement format: MBR=*member
CMNPLPKG	Input for a list of BIND PACKAGE members in the change package. Member statement format: MBR=*member
CMNPLDBR	Input for a list of DBRM members in the change package. Member statement format: MBR=*member
DBBSSTG	Input for the BIND PLAN staging library.
DBBSBAS	Input for the BIND PLAN baseline, production, or temporary install library.
PKGSSTG	Input for the BIND PACKAGE staging library
PKGSBAS	Input for the BIND PACKAGE baseline, production, or temporary install library.
stssysBCTL	Output sequential file containing BIND commands for promotion or permanent installs

DDNAME	Description
stssysTMP	Output PDS library containing BIND command members. This library is not used by delivered Db2 Option functions.

## CMNDB2PL Operation

This section describes how Plan Lookup program CMNDB2PL operates and the factors and variables that can influence the way it works when building bind control for a Db2 Option logical subsystem. The sequence of operations presented here is not exactly the order that the program performs every function.

1. The control statements at ddname CMNPLCTL are scanned to validate the syntax. If an error is found, the program ends with RC=12, and this message is displayed: CMN7005I Control Card Error
2. BIND PLAN members listed in ddname CMNPLDBB and BIND PACKAGE members listed in ddname CMNPLPKG are added to “plans to be bound” list. Each entry in this list is flagged with the assigned library type.
3. CMNDB2PL attempts to add BIND PLAN member names and BIND PACKAGE member names to the “plans to be bound” list for all DBRM members listed in ddname CMNPLDBR. This process is traced for one DBRM to make it easier to understand.
  - a. If control statement USEDB2PACKAGE is present, program CMNDB2PL queries the Db2 catalog table SYSIBM.SYSPACKAGE to find packages that reference the DBRM. If a package name is returned from the query, the program name is matched against member names in libraries concatenated at ddnames PKGSSTG and PKGSBAS. If a match is found, the program name is added to the “plans to be bound list”.
  - b. If control statement NODB2PLAN is omitted or commented out, program CMNDB2PL queries the Db2 catalog table SYSIBM.SYSDBRM to find plans that reference the DBRM. The SYSIBM.SYSDBRM query returns the names of all plans referencing the DBRM. However, CMNDB2PL only wants those plans from the Db2 catalog that are associated with the target logical subsystem. It applies the template for the target logical subsystem to the member names in libraries concatenated at ddnames DBBSSTG and DBBSBAS. If a templated plan name matches a plan name returned from the Db2 catalog query, the untemplated plan name is added to the “plans to be bound” list.
  - c. If no BIND PACKAGE member name or templated BIND PLAN name matches the names returned from the Db2 catalog scans, the name of the DBRM is added to a “bind statement required list”.

4. If the “plans to be bound” list is empty, and control statement NODBRMFOUND is omitted or commented out, CMNDB2PL stops with a RC=12 and the message, NO PLANS HAVE BEEN FOUND TO BIND is printed.
5. The output ddname *stssysBCTL* for the sequential BIND command file is opened and command DSN SYSTEM(*stssys*) is written as the first record.
6. If the “plans to be bound” list is not empty, duplicates are removed from the list, and each remaining name in the list is processed individually. The libraries concatenated at ddnames DBBSSTG and DBBSBAS or at PKGSSTG and PKGSBAS are searched for the first member name that matches a name on the “plans to be bound list”.
  - a. If a match is found, the BIND command member is read, and templates for the target logical subsystem are applied. Unless control statement USEREXIT=(NONE) is input, a call is made to exit program CMNEX101, where user written functions can further modify the BIND command. The modified BIND command is written to the sequential file at ddname *sysBCTL* to be used in a subsequent bind job step.
  - b. If no match is found, and if control statements CREATECC and USEREXIT are input to CMNDB2PL, exit program CMNEX101 is called to execute user written functions to create an appropriate BIND command. The new BIND command is written to the sequential file at ddname *stssysBCTL* to be used in a subsequent bind job step.
  - c. If no match is found, and control statement CREATECC is not input to CMNDB2PL, the name is added to a “not found list”, and at the end processing, CMNDB2PL displays this list as “Bind statements required.” If a BIND member is not found for promote or install, a return code 12 is issued with the message:  
  
CMN7011I CMNDB2PL is terminating due to errors
  - d. If a member is not found for demote or backout, and the Db2 source program is not “new”, a return code 12 is issued unless a different return code is specified on control statement NOBASEDBBRC.
7. An END command is written to ddname *stssysBCTL* as the last records and the file is closed. This file of BIND commands is passed to a job step the performs the binds.
8. The file at ddname *stssysBCTL* is scanned to ensure that every staged DBRM is referenced by a MEMBER statement in a BIND command. If there is a DBRM that is not referenced, CMNDBRPL terminates with RC=12 and issues this message:  
  
CMN7034A Staged DBRM {dbrm name} is not referenced by any plans  
  
You can suppress this return code and error message by using CMNDB2PL control statement IGNORENODBRM.

## BIND Command Keyword Option Order

### Important

BIND Command Keyword Option Order CMNDB2PL uses IBM service routine IKJPARS to parse BIND commands. This ensures that Db2 Option processing is synchronized with changes that IBM might make to BIND keyword operands.

IKJPARS does not attempt to maintain the input order of keyword operands that it parses. Therefore, keyword operands in a BIND command that CMNDB2PL sends to exit programs CMNEX101, CMNEX103, and finally to IKJEFT01 may be in a different order than in the original BIND command member in a staging, promotion, baseline, or production library.

## Keyword Control Statements

---

The following figure shows a sample control statement input to program CMNDB2PL. Detailed descriptions of each keyword are in the sections that follow.



```
//CMNPLCTL DD \*
TYPE=INSTALL
*AUTHORITY=OWNER, INSERT
AUTHORITY=OWNER
*INSERTQUAL
*EARLYCHECK
*IGNORENOSUBSYS
*TEMPDS
*TRACE
USEREXIT=(ASM,NOUNLOAD)
USERID=USER239
PACKAGE=GENL000018
PROJECT=GENL
NOBASEDBBRC=12
WARNINGRC=4
USEDDB2PACKAGE
*NODB2PLAN
*FREEPLAN
*CREATECC
*IGNORENODBRM
*PKLTEMPLATE
DB2ID=C105
LOGICAL=SERT4
PLANTGT=
PLANSRC=
PKGETGT=
PKGESRC=
LOCNTGT=
LOCNSRC=
QUALIFIER=
QUALTGT=
QUALSRC=
OWNER=
OWNRTGT=
OWNRSRC=
REMOTEID=SERT4
```

## Control Statement Syntax

An asterisk in the first position of a program level control statement record disables the statement (comments it out). If a control statement is disabled, the action opposite from that expressed in the keyword name (or defined in this chapter) is in effect.

## Program Level Control Statements

These keyword control statements are input once for each execution of Plan Lookup program CMNDB2PL. The options controlled by these keywords are in effect for all components and logical subsystems processed by the program.

### AUTHORITY=

Specifies whether BIND authorization is by OWNER ID or JOB card. If this control statement is omitted, the default is: AUTHORITY=OWNER

Option	Definition
**OWNER	<p>Required either when the SERNET instance userid has been granted SYSADM or SYSCTRL authority, or when the SERNET userid has one or more secondary authorization IDs, one for each possible owner that can be specified.</p> <p>Each set of logical Db2 subsystem control statements for CMNDB2PL must include an OWNER= control statement, even if the owner ID is left blank.</p> <p>Add an OWNER option to a BIND command if one is not present.</p>
**OWNER, INSERT	<p>If INSERT is not specified, OWNER templating is performed only if an OWNER option is included in the BIND command.</p> <p>JOBID authorization is used when the ChangeMan started task is authorized to submit jobs for other users through USER= field on the JOB card.</p>
**JOBID	<p>No OWNER= control statements can be included in the input at ddname CMNPLCTL.</p>

### BINDPACKAGETYPE

This control statement is obsolete with ZMF 7.

### CREATECC

Sets an indicator for exit program CMNEX101 to create a BIND command when none is found in staging or baseline libraries. The new BIND command is written to ddname stssysBCTL. See the source code for MNEX101.

### EARLYCHECK

Validate the connection to all Db2 subsystems at beginning of processing.

CMNDB2PL builds an internal table that contains all of the Db2 subsystem IDs that will be processed for the job. When EARLYCHECK is specified, CMNDB2PL performs an "early" Db2 connect for each Db2 subsystem ID contained in this table. The NOEARLYCHECK keyword is no longer valid.

## **FREEPLAN**

Create a FREE PLAN command if a BIND PLAN command member cannot be found and CMNDB2PL control statement TYPE= is DEMOTE or BACKOUT.

## **IGNORENODBRM**

Bypass RC=12 and error message "CMN7034A Staged DBRM {dbrm name} is not referenced by any plans." ...when a staged DBRM cannot be matched with a MEMBER statement in the output BIND command file at ddname stssysBCTL.

Issue warning message.

## **IGNORENOSUBSYS**

Bypass RC=12 and error message "CMN7025A Unable to establish connection to Db2 subsystem: {subsystem id}."... ..when CMNDB2PL cannot connect to a Db2 subsystem. Skip all processing for the Db2 subsystem and go on to the next one.

## **INSERTQUAL**

Add a QUALIFIER parameter to a BIND command if one is not present and the QUALIFIER= control statement is not blank.

## **LEGACYCOMMENTS=YES/NO**

Allow comments in the bind component to be indicated by an asterisk in column 1. Default is NO.

The default prevents the possibility of losing the third collection in (e.g.) the following set of bind parameters:

```
BIND PLAN (WILDLOC) PKLIST -
(COLLID1.*,      -
 COLLID2.*,      -
 *.COLLID3.*,   -
 COLLID4.* )
```

If YES then .COLLID3. is treated as a comment. See the Note below.

## **NOBASEDBBRC=**

Specifies the return code that CMNDB2PL issues when processing a new DBRM where no previous BIND command member is found. Valid values are numeric digits (00-99) representing the return code to be issued. The default return code is 04 if this control statement is not present.

## **NOCATALOGDUPCHECK**

Directs CMNDB2PL to bypass duplicate bind checking in a run that relies completely on Db2 catalog driven binds (i.e. there are no plan or package bind control members being processed, just DBRMs). The usual process is to eliminate all duplicate binds generated by the Db2 catalog information. If you prefer to post-process this list of duplicate binds, you can use this keyword to direct CMNDB2PL to bypass duplicate bind checking.

## **NODB2PLAN**

Bypass the query to SYSIBM.SYSDBRM table to determine if any staged DBRMs are referenced by Db2 plans. Use this control statement if plan binds are executed outside of ChangeMan ZMF and only package bind processing is managed by CMNDB2PL.

#### PACKAGE=

Specifies the change package ID. The change package ID is not used by CMNDB2PL but it is passed to exit program CMNEX101.

#### PKLTEMPLATE

Enables templating for the collection IDs and location in the PKLIST of BIND PLAN commands. Without this control statement, Db2 package names will be templated in BIND PACKAGE commands, but the collection IDs and location in BIND PLAN will not be templated.

#### PROJECT=

Application mnemonic.

#### TEMPDS

Write templated BIND command members to the library at ddname stssystem when TYPE=BACKOUT or TYPE=INSTALL.

The NOTMP keyword is no longer valid.

#### TEMPDSNHLQ

High level qualifier for the named temporary dataset allocated to hold DSN BIND records before parsing. The high level qualifier returned by CMNEXINS is used if this keyword is not specified.

#### TRACE

Turn on a trace facility to print diagnostics at ddname SERPRINT from CMNDB2PL calls to CMNDB2CB and CMNDB2SQ for Db2 SQL calls.

#### TYPE=

Indicates the type of operation for which BIND commands are being constructed.

TYPE= must be the first control statement at ddname CMNPLCTL. There is only one TYPE control statement for an execution of CMNDB2PL.

Valid values:

PROMOTE

DEMOTE

INSTALL

BACKOUT

STAGE is accepted, but it processes the same as PROMOTE.

#### USEDDB2PACKAGE

Query Db2 table SYSIBM.SYSPACKAGE to determine if any staged DBRMs are referenced by Db2 packages so that BIND PACKAGE commands can be built. Comment out this control keyword to bypass the query.

#### USEREXIT=

Specifies the language and load mode for exit program CMNEX101.

Format: USEREXIT=(*language,mode*)

Valid values for *language*:

ASM (assembly language) is the only valid language.

Valid values for *mode*:

UNLOAD: Load a fresh copy of the exit load after each invocation when your program is not reusable.

NOUNLOAD: Leave the program resident after the initial LOAD of the user exit. This is the default mode.

USEREXIT=(NONE) bypasses parsing for CMNEX101. You may use this form of the control statement To enhance the efficiency of CMNDB2PL if you do not used CMNEX101.

#### USERID=

Identifies the authorization ID (TSO userid) of the person who issued a promotion or demotion request. For install or backout, USERID identifies the authorization ID of the person who generated the install JCL by freezing the package or issuing a request to rebuild the install JCL from freeze or approval.

Not used by CMNDB2PL but passed to exit program CMNEX101.

#### WARNINGRC=

Specifies the return code CMNDB2PL issues when it issues warnings for taking default actions. The default for warnings is RC=04.

Valid values are numeric digits representing the desired return code to be issued.

#### Note

CMNDB2PL has always assumed that an asterisk in column 1 means a comment card in the bind parameter member. This is not IBM-standard but was (and is) a mechanism used to communicate to the ZMF exit CMNEX101. This, however, meant that genuine bind parameters with an asterisk in column 1 are also treated as comments. We recommend not relying on an asterisk in column 1 of the bind parameters to indicate a comment but, if your processes need to do this, then LEGACYCOMMENTS=YES will allow you to do so (but don't code genuine bind parameters with an asterisk in column 1).

## Logical Subsystem Level Control Statements

These keywords specify values that are defined for the target logical subsystem.

The values for the LOGICAL=, DB2ID=, REMOTEID= statements are defined on the **Db2 Logical Subsystem** panel in application administration for the Db2 Option. The values for the rest of the control statements listed in this section are defined on the **Db2 Logical Subsystem nickname Templates** panel.

If the **Db2 Active Library List** panel for an application directs program CMNDB2PL to process multiple logical subsystems, then a set of these control statements is input to CMNDB2PL for each logical subsystem that is processed.

Control Keyword	Description
DB2ID=	Name of the physical subsystem.

<b>Control Keyword</b>	<b>Description</b>
LOCNSRC=	Source template for PACKAGE location ID.
LOCNTGT=	Target template for PACKAGE location ID.
LOGICAL=	Name (nickname) of the logical subsystem. Not used by CMNDB2PL but passed to exit program CMNEX101.
OWNER=	Insert value for OWNER. Required if AUTHORITY=OWNER or AUTHORITY=OWNER,INSERT. Prohibited if AUTHORITY=JOBID.
OWNRSRC=	Source template for OWNER.
OWNRTGT=	Target template for OWNER.
PKGESRC=	Source template for PACKAGE collection ID.
PKGETGT=	Target template for PACKAGE collection ID.
PLANSRC=	Source template for PLAN name.
PLANTGT=	Target template for PLAN name.
QUALIFIER=	Insert value for QUALIFIER.
QUALSRC=	Source template for QUALIFIER.
QUALTGT=	Target template for QUALIFIER.
REMOTEID=	Site for the physical subsystem. The site may be a local or remote site. Not used by CMNDB2PL but passed to exit program CMNEX101.

## How CMNDB2PL Relates to ChangeMan ZMF

This table shows when CMNDB2PL performs certain operations in the ChangeMan ZMF life cycle. Some of these operations are controlled by keyword control statements. See [Keyword Control Statements](#).

<b>TYPE</b>	<b>Verify BIND member exists in baseline</b>	<b>Read stage BIND command members</b>	<b>Write templated BIND command to PDS</b>
PROMOTE	Yes, if not in stage	Yes	No
DEMOTE	Yes	No	No

<b>TYPE</b>	<b>Verify BIND member exists in baseline</b>	<b>Read stage BIND command members</b>	<b>Write templated BIND command to PDS</b>
INSTALL	Yes, if not in stage	Yes	Yes
BACKOUT	Yes	No	Yes

## CMNDB2PL Return Codes and Messages

This table shows the return codes issues by program CMNDB2PL:

<b>Return Code</b>	<b>Description</b>
RC=00	No errors are detected and no warnings are issued.
RC=04	Warning is issued. Warnings can usually be ignored, but care must be taken with warnings about default actions. Example 1: If no AUTHORITY= card is specified in the control statements, AUTHORITY=OWNER is assumed. Example 2: If no SERNET instance subsystem ID is passed as a program parameter, then CMNDB2PL assumes a "null" subsystem and issues the warning return code.
RC=12	Severe error, processing is halted. ## Modify Return Codes

Use these control statements allow the bind step to process when CMNDB2PL encounters a problem. See [Program Level Control Statements](#).

These keyword control statements modify the return code issued by CMNDB2PL.

- NOBASEDBBRC=
- WARNINGRC=

These control statements bypass error conditions:

- IGNORENODBRM
- IGNORENOSUBSYS

## Messages

All messages issued by plan lookup program CMNDB2PL are explained in the *ChangeMan ZMF Messages Guide*.

## Sample CMNDB2PL Report

The following is an annotated sample report from CMNDB2PL: This section of the report shows the plan being used for Db2 queries by CMNDB2PL.

```
ChangeMan(R) ZMF DB2 Option Plan Lookup Program WEDNESDAY FEBRUARY 3, 2016 (2016/034) 01:35:12
CMNDB2PL - 8.1.0 10/10/2014 11.24

DB2 Plan Used by ChangeMan ZMF Call Attach Facility:
Using plan (CMNPLAN)
```

The next section prints copies of the control card input. Messages for any errors found in the input records are interspersed with the control cards. An error message immediately follows the control card that caused the error.

```
Control card input (DDNAME = CMNPLCTL)
```

```
1 ==> TYPE=INSTALL
2 ==> AUTHORITY=OWNER
3 ==> USEREXIT=(ASM,NOUNLOAD)
4 ==> USERID=USER239
5 ==> PACKAGE=ACTP000084
6 ==> PROJECT=ACTP
7 ==> NOBASEDBBRC=12
8 ==> WARNINGRC=4
9 ==> USEDDB2PACKAGE
10 ==> DB2ID=C105
11 ==> LOGICAL=SERT7
12 ==> PLANTGT=
13 ==> PLANSRC=
14 ==> PKGETGT=
15 ==> PKGESRC=
16 ==> LOCNTGT=
17 ==> LOCNSRC=
18 ==> QUALIFIER=
19 ==> QUALTGT=
20 ==> QUALSRC=
21 ==> OWNER=
22 ==> OWNRTGT=
23 ==> OWNRSRC=
24 ==> REMOTEID=SERT7
```

This section contains a list of all BIND PACKAGE and BIND PLAN members in this change package. These BIND commands are passed to the bind utility after the templates have been applied.



Staged bind control statements in this change package (DDNAME = CMNPLPKG)

1 ==> MBR=ACPSRCD1

This section contains a list of all DBRMs in this change package. This list is used when the query is built to find the list of all plans in the affected Db2 catalogs that reference these DBRMs.

Staged DBRMs in this change package (DDNAME = CMNPLDBR)

1 ==> MBR=ACPSRCD1

Since NODB2PLAN was specified, the Db2 catalog query for existing DBRMs is not executed. If NODB2PLAN had been commented out, a query for existing DBRM's would be issued at this point.

The required BIND command members are listed. The origin of the BIND command requirement is one of the following:

A staged BIND PLAN member, indicated by DD: DBBSSTG.

A staged BIND PACKAGE member, indicated by DD *pkgbind*SSTG.

A plan that was found in the Db2 catalog that contained one of the staged DBRMs, indicated by DB2:SYSDBRM.

A package found in SYSIBM.SYSPACKAGE, indicated by DB2:SYSPACKAGE.

The following bind control statements are required:

Origin of Location	Staged Owner	Actual Name	Logical Qualifier	Plan DB2	Plan Remote	Package Reject
Bind Reqmt. Template	Name	Name	Subsys	Template	Template	Template
	Template	Template	Template	Subs	ID	
DD:PKGSSTG	GNLSDB01	GNLSDB01				
S3P1UT		????????????????		????????????????	????????	????????
C105	SERT3P1					

If no BIND command members are found for any plan, CMNDB2PL terminates with a return code of 12, and lists all such DBRMs.

## Secondary Binding

---

Secondary binding uses CMNDB2PL to produce bind parameters for logical subsystems other than the primary logical subsystem (i.e. the target of the promote etc. action).

In addition to generating the relevant bind parameters CMNDB2PL (for TYPERUN=PROMOTE or INSTALL) will check consistency tokens in the Db2 catalog to see if the secondary bind is actually needed.

This functionality is affected by the following, secondary bind only, CMNPLCTL parameters.

<b>Control Keyword</b>	<b>Description</b>
SECONDARYBIND	Requests secondary bind processing for this execution.
PKG2TGT=	The target template for the primary logical subsystem PACKAGE collection id.
PKG2SRC=	The source template for the primary logical subsystem PACKAGE collection id.

All the other template keywords refer to the requirements of the secondary logical subsystem.

PKG2TGT/SRC are required to allow CMNDB2PL to form the appropriate collection id in order to query the catalog for the latest consistency token for the bind that has just taken place for the primary logical subsystem.

If this consistency token is already present for the secondary collection id then no bind is required.

# 8. Stored Procedure Utilities

---

This chapter discusses the stored procedure utilities available with the Db2 Option.

- [Introduction](#)
- [CMNDB2AV](#)
- [CMNDB2DD](#)
- [CMNDB2SL](#)
- [CMNDB2TR](#)
- [CMNDB2DR](#)
- [Stored Procedure Walkthrough](#)

## Introduction

---

This table lists the utilities that support management of Db2 stored procedures, triggers, and user defined functions in the Db2 Option.

<b>Program Name</b>	<b>Functional Description</b>	<b>Library Sub-type</b>	<b>Type of Component</b>
CMNDB2AV	Activate SP versions	N	Native SQL stored procedure
CMNDB2DQ	Extract CREATE SQL from SQL stored procedure source. Pass SQL to utility CMNDB2DD to define the stored procedure.	Q	SQL stored procedure
CMNDB2DD	Execute CREATE SQL statements to register stored procedures, user defined functions, and triggers in a Db2 catalog.	D N T (Q)	CREATE SQL statement

<b>Program Name</b>	<b>Functional Description</b>	<b>Library Sub-type</b>	<b>Type of Component</b>
CMNDB2SL	Recycle stored procedures and external user defined functions in a WLM-managed address space.	S	Stored procedure and external user defined function load modules, REXX stored procedure.
CMNDB2TR	Drop and recreate triggers to maintain the current firing order when a trigger for the same table/event/time is changed or added.	T	Trigger
CMNDB2DR	Report CREATE PROCEDURE and CREATE FUNCTION in a change package that have dependencies that will interfere with the DROP automatically issued by the Db2 Option before the CREATE is executed.	D Q	CREATE SQL

## CMNDB2AV

This program is designed to aid the automation of the activation of various versions of the same stored procedure. It is driven by SYSIN parameters (described below) which are, normally, prepared by CMNDB2DD. CMNDB2DD passes these parameters to a subsequent job step which submits a new job to activate the new version of the SP.

CMNDB2AV requires the presence of the CMNZMF.CMNDB2\_ATTRIBS table in the target Db2 subsystem. It also requires the CMNDB2AT package to be bound using the CMNZMF collection id.

Here is a description of the sysin parameters followed by the actions taken for each function type.

<b>Parameter</b>	<b>Description</b>
FUN=ADDNEW / REACTIVATE / BACKOUT	Specifies which function is being performed. If FUN= is not set to one of these three values then CMNDB2AV will end without doing anything. Specifically, if CMNDB2DD has decided there is nothing to do it will pass FUN=DO_NOTHING to this program.

Parameter	Description
ZMF=CMNx	This specifies the ZMF subsystem under whose direction the current action is taking place. Note that it is possible that the same Db2 subsystem could be targeted by more than one ZMF. This value ensures that rows written to the CMNDB2_ATTRIBS table are unique to each ZMF.
PKG=aaaannnnnn	The package associated with the current action.
DB2=ssss	The Db2 subsystem which is the target of this action.
SCM=	The schema of the SP whose version is to be activated.
NAM=	The name of the SP whose version is being activated.
VER=	The SP version. This has different meanings depending on the FUN= type. E.g. for ADDNEW it is the version to be activated. For REACTIVATE it indicates the version of the SP which is being deactivated. For BACKOUT it is ignored (and not generated by CMNDB2DD).

Note the last three parameters may be longer than will fit on one 80 byte card image. If this is the case then a non-blank character (which is ignored) is placed in col 80 and the value is continued from the start of the next card image. E.g.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
SCM=AREALLY8101234567820123456783012345678401234567850123456786012345678701234R+
EALLYLONGSCHEMA

```

What each function type does:

Function	Description
ADDNEW	Extracts the current active version for the SP from the target Db2 catalog. Writes a row to CMNDB2_ATTRIBS to note this version was the 'prior active' version for the SP. Issues the activation request for the new version as indicated by the VER= sysin parameter.

Function	Description
BACKOUT	1) Gets the current active version of the SP from the Db2 catalog. 2) Extracts the recorded 'prior version from the CMNDB2_ATTRIBS row for this zmfid/zmf package/location/schema/SP name. 3) Issues the activation request for the 'prior version' as obtained in 2). 4) Drops the version obtained in 1). 5) Removes the CMNDB2_ATTRIBS row for this zmfid/zmf package/location/schema/SP name.

## CMNDB2DQ

Program CMNDB2DQ runs at promote, demote, install, and backout. It dynamically calls the Db2 precompiler to extract the SQL required to define an object to Db2, removing any procedural code within the CREATE object definition. It can be used for any SQL component type, but program CMNDB2DQ is required when processing SQL language stored procedures.

The CREATE SQL extracted by this program is written to a file that is passed to utility program CMNDB2DD where the SQL is executed to register the Db2 objects in Db2 catalogs.

This table shows the skeletons that include program CMNDB2DQ and where those skeletons are used.

Skeleton	...embedded in skeletons	Skeleton Description
CMN\$ \$PSQ	CMN\$\$PRM	Perform promotion or demotion to local sites.
	CMNIMPRM	IMS Option: Perform promotion or demotion to local sites.
CMN\$ \$RSQ	CMNRPICR	IMS Option: Perform remote promotion or demotion.
	CMNRPMCR	Perform promotion or demotion to remote sites.
CMN\$ \$SQL	CMN21	Used to perform db2 binds and/or ddl processing for installation of packages into production libraries and db2 catalogs.

<b>Skeleton</b>	<b>...embedded in skeletons</b>	<b>Skeleton Description</b>
	CMN49	Used to perform db2 binds and/or ddl processing for backout of packages from production libraries and db2 catalogs.

## Keyword Control Statements

This table describes the keyword options that control the behavior of program CMNDB2DQ.

<b>Keyword</b>	<b>Description</b>
NOTFOUNDRC=	Return code set if the requested SQL component (MBR=) is not found in the library concatenation at ddname SQLIN. Valid values: 0 to 99 Default: 8 Comment: Set to 0 for demotion and backout functions.
MBR=	Name of SQL component to be processed. The number of MBR control statements is not limited.

## Return Codes and Messages

Program CMNDB2DQ calls the Db2 precompiler. The return codes from the precompiler are passed through as CMNDB2DQ return codes.

## CMNDB2DD

Program CMNDB2DD executes DDL/SQL at promote, demote, install and backout to register stored procedures, triggers, and user defined functions in the Db2 catalog.

The program reads a specified member from a concatenation of SQL libraries, parses the records in the member into SQL sentences using a specified terminator, applies all relevant templates and offers the SQL sentences to the target Db2 subsystem. Further manipulation of the DDL/SQL may be achieved using general token templates and/or the HLLX exit facility. The results of the execution may also, optionally, be passed in an output dataset to be processed by some other Db2 utility of your choosing.

Program CMNDB2DD writes to ddname TRIGGER the tablename/event combinations for which a trigger has been added or updated. This file is passed to utility program

CMNDB2TR.

This table shows the skeletons that include program CMNDB2DD and where those skeletons are used.

<b>Skeleton</b>	<b>...embedded in skeletons</b>	<b>Skeleton Description</b>
CMN\$\$PSQ	CMN\$\$PRM	Perform promotion or demotion to local sites.
	CMNIMPRM	IMS Option: Perform promotion or demotion to local sites.
CMN\$\$RSQ	CMNRPICR	IMS Option: Perform remote promotion or demotion.
	CMNRPMCR	Perform promotion or demotion to remote sites.
CMN\$\$SQL	CMN21	Used to perform db2 binds and/or ddl processing for installation of packages into production libraries and db2 catalogs.
	CMN49	Used to perform db2 binds and/or ddl processing for backout of packages from production libraries and db2 catalogs.

## Keyword Options

This list describes the keyword options that control the behavior or program CMNDB2DD. Default values are in italics:

**ACTION=PROMOTE/DEMOTE/INSTALL/BACKOUT** This setting is used by the Native SQL SP version process. See description of keyword SPVERPKGRC. The value is also passed to the HLL exit if it is active

**AUTODROP=YES/NO** Autodrop facility issues a DROP command for a procedure, trigger, or user defined function before processing CREATE SQL for the object.

**BINDDEPLOY=** Specify any extra clauses to be included in the generated BIND DEPLOY command for a native SQL stored procedure. The value for this keyword is freeform text and whatever you put here will be appended to the command as is. You may specify as many BINDDEPLOY= keywords as you wish. An example might be BINDDEPLOY=QUERYACCELERATION(ENABLE).

**CMP=** Synonym for MBR=. See MBR=.

**DB2ID=** Db2 subsystem ID to which SQL should be presented. The DB2ID keyword control statement must precede one or more MBR keyword control statements.

**DEPLOYFROMLOCATION=** This is the Db2 location to which CMNDB2DD will route the call to ADMIN\_COMMAND\_DSN in order to execute the BIND DEPLOY request. The value is populated from the 'Deploy' value for location on the source logical subsystem definition.



**DEPLOYQUAL=**

**DEPLOYOWNER=** If standard templating processes result in a blank value for qualifier and/or owner when a bind deploy command has been requested then any values entered for these parameters will be used on the command (cf. bind insert values). Values for both the templates and these 'deploy' fields will be provided by the target logical subsystem definition.

**DROPRC=** Return code set if the requested SQL action is DROP and the component is not found in the Db2 catalog.

Valid values: 0 to 99

Default: 0

**ERRSTOPAFT=** Number of SQL errors allowed before the program is terminated.

Valid values: 0 to 99999999

Default: 0

**HLLX=(name,type)** This parameter indicates that a HLL exit be taken by CMNDB2DD. This does not use the standard HLLX scheduling system as CMNDB2DD needs to be able to run on remote z/OS images. However, the call mechanism is the same, i.e. the exit can be coded in REXX or any LEsupported language.

The purpose is to allow you to manipulate the DDL being processed by CMNDB2DD directly using your own business logic. It will also allow them to stop CMNDB2DD from continuing should it decide to do so.

The 'name' sub-parameter specifies the external name of the HLL exit (i.e. the REXX exec name of LE program name). The 'type' sub-parameter must be either REXX or LE as appropriate.

**LINEFEED=YES/NO** If this is set to YES then CMNDB2DD will insert a linefeed character (EBCDIC x'25') to the end of each physical line of SQL code. This is useful during formatting by various debug tools. Default is NO. Note that Data Studio inserts its own linefeed characters

**MBR=** Name of SQL component to be processed in the Db2 subsystem specified in the preceding DB2ID keyword control statement.

**NOTFOUNDRC=** Return code set if the requested SQL component (MBR=) is not found in the library concatenation at ddname SQLIN.

Valid values: 0 to 99

Default: 8

Comment: Set to 0 for demotion and backout functions.

**PASSTHRU=YES/NO** The usual method that CMNDB2DD employs is to present DROP/CREATE DDL directly to the target Db2 subsystem. Customers may wish to do this using other utilities but may also wish to avail themselves of the facilities offered by CMNDB2DD. This new parameter allows them to do this. If PASSTHRU=YES is specified then the resulting DDL, as manipulated by this program, is then written to an output ddname rather than being presented to Db2. The ddname used is:

```
//ssssOUT
```

Where ssss is the Db2 subsystem id currently being processed (i.e. as directed by the DB2ID= sysin parameter). The output DCB is checked for compatibility with the SQLIN ddname. The member read from SQLIN is directed by the MBR= sysin parameter and this same member name is used to write to ssssOUT.

The following message will be seen in sysprint:

```
CMNDD033I Sentence passed to ssssOUT, no action taken at target Db2 subsystem. The ssss is resolved to the actual subsystem id in the message.
```

**SPVERPKGRC=n** 8 is the default. This is the return code set should the ZMF id and package name check, described in the keyword ACTION for demote/backout, fail.

**SPVERSION= NO YES ONLY COMMAND BOTH UNDO** Lets CMNDB2DD know whether Native SQL SP versioning is supported.

**YES:** As well as the standard templating/presentation of the SP SQL to Db2 we also take actions designed to activate the correct SP version at the target Db2 subsystem. This is done by writing transactions (intended for CMNDB2AV) to the VERSION DDname. In the supplied skeletons this ddname is passed to a subsequent job submission step which submits an execution of CMNDB2AV to act on these transactions

**ONLY:** Do not present SQL to Db2, extract version information and write to the VERSION ddname only.

**COMMAND:** Generate the relevant BIND DEPLOY command and call location.SYSPROC.ADMIN\_COMMAND\_DSN to execute it.

**BOTH:** Equivalent to ONLY and COMMAND - take both actions.

**UNDO:** This takes the same action as ONLY but uses the COLLID templates to generate the schema of the SP whose version information we need (ONLY uses the SCHEMA templates). This action is required during demote and backout.

**SQLTERM=** Alternate SQL statement terminator. If the input includes SQL that uses the semicolon (;) as a statement terminator, specify an alternate terminator for the input so that the semicolon is passed through to the server.

You can specify any character except the following:

- blank
- comma
- underscore
- single quote
- double quote
- left parenthesis
- right parenthesis

If you omit this keyword parameter, the default SQL statement terminator is semicolon (;).

**SQUEEZE=YES/NO** If this is set to YES then superfluous blanks will be stripped out from the SQL sentence prior to it being presented to Db2. YES is the default for legacy reasons. If you wish to be able to view the Db2 object back directly from the Db2 catalog using a debugger or some other tool then you will want to use SQUEEZE=NO.

**SRCCOLLIDTEMPLATE=** Source template for COLLID parameter in SQL definitions for stored procedures and user defined functions.

**SRCQUALTEMPLATE=** Source template for explicit table qualifier in SQL definitions.

**SRCSCHEMATEMPLATE=** Source template for schema in SQL definitions.

**SRCWLMTEMPLATE=** Source template for WLM application environment.

**TEST=** Program trace facility.

Valid values:

YES/NO

Default: NO

**TGTCOLLIDTEMPLATE=** Target template for COLLID parameter in SQL definitions for stored procedures and user defined functions.

**TGTQUALTEMPLATE=** Target template for explicit table qualifier in SQL definitions.

**TGTSCHEMATEMPLATE=** Target template for schema in SQL definitions.

**TGTWLMTEMPLATE=** Target template for WLM application environment.

**TOKENNAME=**

**TOKENSRCT=**

**TOKENTGTT=** The parameters allow you to define your own DDL parameters to be template via 'standard' CMNDB2DD templating. Freeform token processing will take place in addition to and after all the existing fixed name clause processing (e.g. after the likes of owner, qualifier etc. templating).

**TOKENNAME=** specifies a string which will be looked for in the DDL. This may include imbedded blanks as long as the whole string is enclosed in single quotes. If there are no imbedded blanks then quotes are optional. Strings including imbedded blanks must not contain more than 5 subwords (and each subword must be 16 bytes).

In the DDL, to be recognized, the token name may be preceded by either a blank or a comma, and followed by a blank or a left hand bracket.

The next word following the token will be templated according to the standard rules with the (optional) source template being supplied via **TOKENSRCT** and the (required) target template by **TOKENTGTT**.

As many of these groups as you need may be specified. They are processed sequentially.

The code applies the template to each found occurrence of **TOKENNAME** (it doesn't stop looking after the first found, only stopping when the current SQL sentence is exhausted). The resulting DDL is then subject to the next set of **TOKEN** templates and so on.

The **TOKENNAME** value may be up to 64 bytes The **TOKEN** template fields may be up to 128 bytes and can be specified across lines as per the other fixed name templates.

**TOLSTDNUM=YES/NO** If this is set to YES then the last 8 bytes of each 'card image' is ignored. YES is the default for legacy reasons, 80 byte card images being the standard format for DDL/SQL. Variable length records have been supported for some time (e.g. Data Studio generated SQL needs VB,255), in anything other than standard 80 byte card images you should use **TOLSTDNUM=NO**.

**TRACKTRIGGER=YES/NO** Write to ddname **TRIGGER** the tablename/event combinations for which a trigger has been added or updated.

**ZMFID=CMNx** This is the ZMF subsystem id which owns the package associated with this action. It is used in the SP version validation process for demote/backout. It is also passed on in the **VERSION** ddname transactions.

**ZMFPACKAGE=aaaannnn nn** This is the ZMF package name associated with this action. It is used in the SP version validation process for demote/ backout as. It is also passed on in the **VERSION** ddname transactions.

See the description of the Db2 Logical Subsystem nickname Templates panel in [Define Global Logical Subsystems](#) for an explanation of Source and Target templates. See [Templating Examples](#) to see how the Source and Target fields interact to modify templated fields in SQL.

A new DB2ID control statement makes program CMNDB2DD disconnect from the current subsystem and connect to the new one. The number of MBR control statements that follow a DB2ID control statement is not limited.

## Return Codes and Messages

Return Code	Description
00	Success
04	Warnings Issued
08	SQL errors
12	Parameter errors
16	Other unrecoverable errors

## CMNDB2SL

Program CMNDB2SL runs at promote, demote, install, and backout to recycle stored procedures and user defined functions that have been changed. This program searches the Db2 catalog for procedures and functions defined on an external (load module) name. It uses the VARY WLM...REFRESH command to recycle these objects in the WLM-managed address space.

This table shows the skeletons that include program CMNDB2SL and where those skeletons are used.

Skeleton	...embedded in skeletons	Skeleton Description
CMN\$ \$PST	CMN\$\$PRM	Perform promotion or demotion to local sites.
	CMNIMPRM	IMS Option: Perform promotion or demotion to local sites.
	CMNRPMCR	Perform promotion or demotion to remote sites.

<b>Skeleton</b>	<b>...embedded in skeletons</b>	<b>Skeleton Description</b>
CMN\$ \$STP	CMN20/CMN20I	Used to perform Db2 binds and/or ddl processing for installation of packages into production libraries and Db2 catalogs.
	CMN50/CMN50I	Used to perform Db2 binds and/or DDL processing for backout of packages from production libraries and Db2 catalogs.

## Keyword Options

This table describes the keyword options that control the behavior of program CMNDB2SL.

<b>Keyword</b>	<b>Description</b>
TEST=	Program trace facility. Valid values: YES/NO Default: NO
DB2ID=	Db2 subsystem ID to which Db2 commands should be presented. The DB2ID keyword control statement must precede one or more MBR keyword control statements.
WLMENVMASK=	WLM application environment mask.
MBR=	External (load module) name of stored procedure or user defined function to be recycled.

A new DB2ID control statement makes program CMNDB2SL disconnect from the current subsystem and connect to the new one. The number of MBR control statements that follow a DB2ID control statement is not limited.

## Return Codes and Messages

<b>Return Code</b>	<b>Description</b>
00	Success
04	Warnings Issued
08	Command errors
12	Parameter errors

Return Code	Description
16	Other unrecoverable errors

## CMNDB2TR

Program CMNDB2TR runs at promote, demote, install, and backout to maintain the firing order of existing triggers when a new trigger is added or an existing trigger is changed by utility program CMNDB2DD.

This program reads a file created by utility program CMNDB2DD that lists CREATE TRIGGER definitions have been executed. Program CMNDB2TR queries SYSIBM.SYSTRIGGERS to see if multiple triggers have been defined for the same table/ event/time combination. If multiple triggers are defined, then triggers with CMNFIRE#*nn* coded on the COMMENT ON field are dropped and recreated in the *nn* sort sequence. All other triggers are then recreated in the original CREATEDTS order.

CREATE TRIGGER SQL executed by CMNDB2TR is built from the Db2 catalog entries.

This table shows the skeletons that include program CMNDB2TR and where those skeletons are used.

Skeleton	...embedded in skeletons	Skeleton Description
CMN\$ \$PSQ	CMN\$\$PRM	Perform promotion or demotion to local sites.
	CMNIMPRM	IMS Option: Perform promotion or demotion to local sites.
CMN\$ \$RSQ	CMNRPICR	IMS Option: Perform remote promotion or demotion.
	CMNRPMCR	Perform promotion or demotion to remote sites.
CMN\$ \$SQL	CMN21	Job to perform Db2 binds and/or DDL processing for installation of packages into production libraries and Db2 catalogs.

<b>Skeleton</b>	<b>...embedded in skeletons</b>	<b>Skeleton Description</b>
	CMN49	Job to perform Db2 binds and/or DDL processing for backout of packages from production libraries and Db2 catalogs.

## Return Codes and Messages

<b>Return Code</b>	<b>Description</b>
00	Success
04	Warnings Issued
08	SQL errors
12	Parameter errors
16	Other unrecoverable errors

## CMNDB2DR

### Db2 Object Dependency Report

The Db2 Object Dependency report is a batch report that analyzes stored procedures and user defined functions for dependencies that will interfere with the automatic DROP that is issued before a CREATE is executed at promote, demote, install, or backout.

Run this report for each package that contains CREATE PROCEDURE or CREATE FUNCTION statements for stored procedures and user defined functions to find potential problems with the automatic DROP that is issued before each CREATE SQL is processed.

Program CMNDB2DR reads specified members in a PDS library of Db2 object definitions. It parses the contents of each member looking for any of:

```
CREATE PROCEDURE
CREATE FUNCTION
```

It extracts the name of the object to be created and reports on any other objects which are dependent on this object.

Program CMNDB2DR can be run in batch using member CMNDB2DR delivered in the CMNZMF CNTL library.



The program can also be initiated online from the **Db2 Object Dependency Report** panel CMNDB20D, which is accessed from:

- The **Define or Generate ChangeMan Batch Reports** panel in global administration at **=A.G.R**, option **5 Db2**
- The **Define or Generate ChangeMan Batch Reports** panel in application administration at **=A.A.R**, option **5 Db2**
- The **Submit ChangeMan Batch Reports** panel at **=6**, option **2 Db2**.

```

CMNDB20D                Db2 Object Dependency Report
Command ==> _____

Package . . . . . ACTP000070
Target Db2 subsystem . . C105

Job Statement Information:

//USER015N JOB (SM-1IKF-SM), 'DB2 OBJECT',
//          CLASS=A,MSGCLASS=X,NOTIFY=USER015
//*
//*
```

This table describes the fields on the **Db2 Object Dependency Report** panel.

Field	Description
Package id	Type the ID of the package that you want to analyze. The package is scanned for Db2 components which may contain user defined function or stored procedure definitions. The batch job will analyze the catalog tables in the target Db2 subsystem to see if any objects exist that are dependent on the components in this package.
Target Db2 subsystem	Type the identifier of the target Db2 subsystem.
Job Statement Information	Specify a valid JOB statement for that batch job that will be submitted when you press Enter.

## Keyword Options

This table describes the keyword options that control the behavior of program CMNDB2DR.

Keyword	Description
TEST=	Program trace facility. Valid values: YES/NO Default: NO
AUTHID=	The userid or qualifier to be used when no explicit schema is provided.
NOTFOUNDRC=	Return code set if the requested SQL component (MBR=) is not found in the library concatenation at ddname SQLIN. Valid values: 0 to 99 Default: 8 Comment: Set to 0 for demotion and backout functions.
INOPRC=	Return code set if the automatic DROP will work but will cause dependent objects to be made inoperative. Valid values: 0 to 99 Default: 8
FAILRC=	Valid values: 0 to 99 Default: 8 Return code set if the automatic DROP will fail.
DB2ID=	Db2 subsystem ID to which SQL should be presented. The DB2ID keyword control statement must precede one or more MBR keyword control statements.
MBR=	Name of SQL component to be processed in the Db2 subsystem specified in the preceding DB2ID keyword control statement.

A new DB2ID control statement makes program CMNDB2DR disconnect from the current subsystem and connect to the new one. The number of MBR control statements that follow a DB2ID control statement is not limited. Report output (excerpts):

ChangeMan(R) ZMF            CMNDB2DR - 8.1.0 DB2 object dependency report

CMNDB2DR            Processing begins at 15:35:29 on 02/04/2016

-----  
CMNDB2DR SYSIN: TEST=NO  
CMNDB2DR SYSIN: AUTHID=JPRESTO  
CMNDB2DR SYSIN: NOTFOUNDRC=4  
CMNDB2DR SYSIN: INOPRC=8  
CMNDB2DR SYSIN: FAILRC=8  
CMNDB2DR SYSIN: DB2ID=S10G

-----  
CMNDR014I            Now connected to DB2 subsystem : C105

-----  
CMNDB2DR SYSIN: MBR=SQLNAT01

CMNDR024I            Member contains procedure : JPRESTO.SQLNAT01  
CMNDR025I            No dependencies found for this object

-----  
CMNDB2DR            Processing completed at 15:35:30 on 02/04/2016 MAX RC = 00

## Return Codes and Messages

Return Code	Description
00	Success
04	Warnings Issued
08	DROP errors
12	Parameter errors
16	Other unrecoverable errors

# Stored Procedure Walkthrough

This section is kept for documentation of legacy processes, see the section on Native SQL stored procedures for more current practices.

Here we show the ZMF Db2 Option stage and promotion processes for SQL stored procedure SQL00002 from the sample used by Db2 Connect (*IBM Db2 Connect User's Guide, SC09-4835*).

```

CMNSTG01 STAGE:                ACTP000072 Components                Row 1 to 1 of 1
Command ==> _____ Scroll ==> CSR

      Name      + Type  Status   Changed      Procname  User    Request
b  SQL00002      SPQ   INCOMP   20160204 202145      JPRESTO  *BUILD
***** Bottom of data *****
  
```

```

ISRBROBA NTP.S6.ACTP.STG6.\#000072.SPQ(SQL00002) - 01 Line 00000000 Col 001 080
Command ==> _____ Scroll ==> CSR
***** Top of Data *****
CREATE PROCEDURE USER15.PROCEDURE2 ( )
  RESULT SETS 1
  LANGUAGE SQL
  EXTERNAL NAME SQL00002
  COLLID TEST
  WLM ENVIRONMENT C105SP
  RUN OPTIONS 'TEST(ALL,*, ,VADTCPIP&192.168.1.3:*)'
P1: BEGIN
  DECLARE cursor1 CURSOR WITH RETURN FOR
    SELECT SCHEMA, NAME FROM SYSIBM.SYSROUTINES;
  OPEN cursor1;
END P1
***** Bottom of Data *****
  
```

Staging stored procedure SQL00002 displays the following panel.

```

CMNSTG04                Stage: Build                HISTORY ASSUMED
Command ==> _____

      Package: ACTP000072      Status: DEV      Install Date: 20160229

Staged name . . . . SQL00002                +
Library type . . . . SPQ - DB2 Stored Proc Source - SQL Language
Dataset name . . . . CMNTP.S6.ACTP.STG6.\#000072.SPQ                +

Language . . . . . SQL      (Blank for list)
Compile procedure . . . . CMNSQL (Blank for list; ? for designated proc.)
Compile parms . . . . . _____
Pgm binder parms . . . . . _____

Enter "/" to select option
 / Db2 processing
 / Other Db2 options
 _ Other options
 _ Suppress messages
  
```

Job statement information:

```
//USER015D JOB (SM-1IKF-SM), 'TEST',
//          CLASS=A,MSGCLASS=X,NOTIFY=JPRESTO
//*
//*
```

Language SQL is associated with compile procedure CMNSQL.

Skeleton CMN\$\$\$CEE is customized to provide STEPLIB data set names.

```
ISRSUPC - MVS/PDF FILE/LINE/WORD/BYTE/SFOR COMPARE UTILITY- ISPF FOR z/OS      2013/01/31 16.08 PAGE 6
NEW: CMNTP.S4.V71201.CMNZMF.CUSTOM.SKELS(CMN$$$CEE)   OLD: CMNTP.S0.V712.CMNZMF.SKELS(CMN$$$CEE)

          LISTING OUTPUT SECTION (LINE COMPARE)

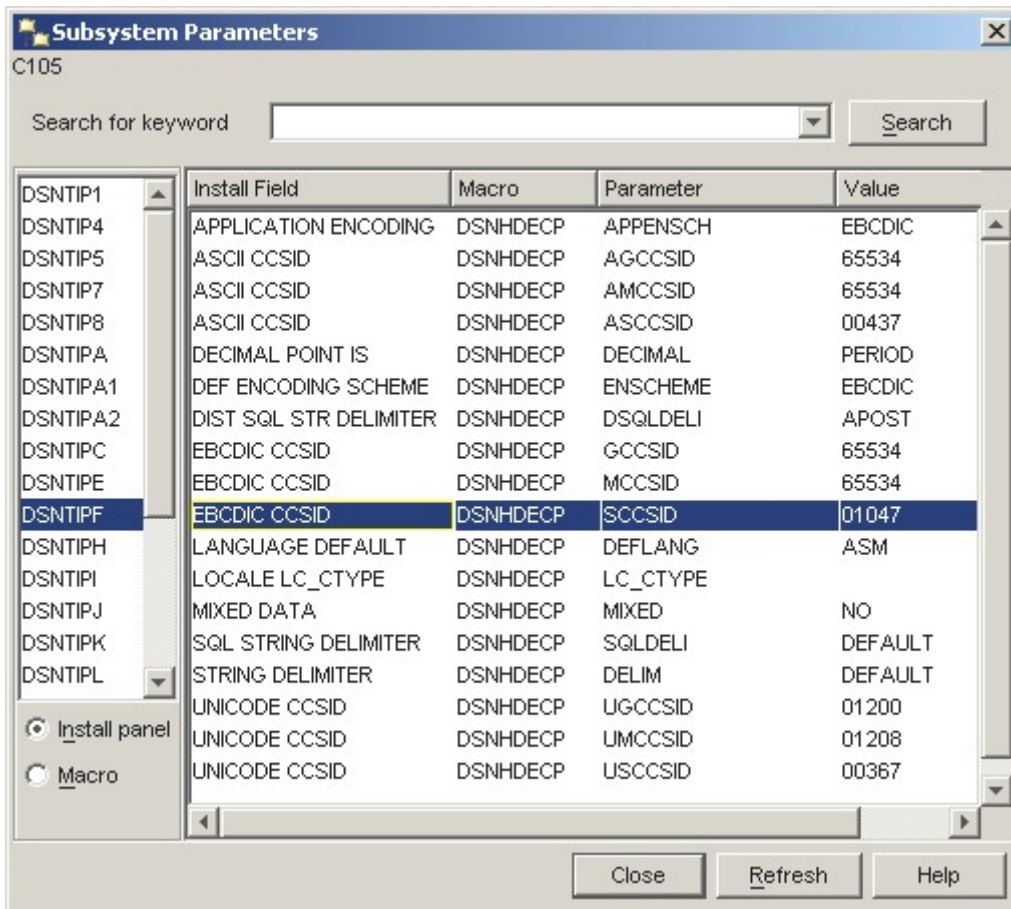
ID      SOURCE LINES                                     TYPE   LEN N-LN#  O-LN#
-----1-----2-----3-----4-----5-----6-----7-----8
//          REGION=0M,                                     00008 00008
//          COND=(4,LT),                                   00009 00009
//          PARM=('&COMPPRM1',                             00010 00010
)SEL &COMPPRM2 NE &Z                                     00011 00011
//          '&COMPPRM2',                                   00012 00012
)ENDSEL &COMPPRM2 NE &Z                                   00013 00013
)SEL &COMPPRM3 NE &Z                                     00014 00014
//          '&COMPPRM3',                                   00015 00015
)ENDSEL &COMPPRM3 NE &Z                                   00016 00016
//          '&COMPOPT')                                     00017 00017
I - //STEPLIB DD DISP=SHR,DSN=CBC.SCCNCMP                 RPL=   2 00018 00018
D - //STEPLIB DD DISP=SHR,DSN=somnode.SCCNCMP
I - //          DD DISP=SHR,DSN=CEE.SCEERUN                00019 00019
D - //          DD DISP=SHR,DSN=somnode.SCEERUN
)IM CMN$$SYC                                             MAT=   1 00020 00020
I - //          DD DISP=SHR,DSN=CEE.SCEEH.H               RPL=   1 00021 00021
D - //          DD DISP=SHR,DSN=somnode.SCEEH.H
)SEL &DB2PC EQ Y                                         MAT=   1 00022 00022
I - //          DD DISP=SHR,DSN=SYS2.DB2810.SDSNC.H       RPL=   1 00023 00023
D - //          DD DISP=SHR,DSN=somnode.SDSNC.H
)ENDSEL &DB2PC EQ Y                                     MAT=  65 00024 00024
)CM
)CM DEFAULT OPTIONS FOR SQL STOREDP PROCEDURE GENERATED C
)CM
//SYSLIN DD DISP=(,PASS),DSN=####OBJECT&C\#N(&CMPNAME),  00028 00028
//          UNIT=SYSDA,SPACE=(CYL,(1,1,1)),              00029 00029
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)         00030 00030
//SYSPRINT DD DISP=(,PASS),DSN=####LIST30C&C\#C,        00031 00031
//          UNIT=&DEFNVUN,SPACE=(CYL,(5,5),RLSE),        00032 00032
//          DCB=(RECFM=FBM,LRECL=133,BLKSIZE=23275)     00033 00033
*** CHANGE SECTION CUTOFF *****
```

Skeleton CMN\$PARM is customized to provide execution parameter CCSID(1047) for the Db2 precompiler in skeleton CMN\$\$PDB. The default CCSID pair for a z/OS 1.8 LE environment is (1047,819), where 1047 indicates the EBCDIC IBM-1047 codepage and 819 indicates the ASCII ISO8859-1 codepage.

LISTING OUTPUT SECTION (LINE COMPARE)

ID	SOURCE LINES	TYPE	LEN	N-LN#	O-LN#
	-----1-----2-----3-----4-----5-----6-----7-----8				
	)SET LINKPRM2 = &Z			00575	00575
	)SET LINKPRM3 = &Z			00576	00576
	)CM			00577	00577
	)ENDSEL &LNGNAME EQ SASC			00578	00578
	)CM			00579	00579
	)CM			00580	00580
	)CM SQL			00581	00581
	)CM			00582	00582
	)SEL &LNGNAME EQ SQL			00583	00583
	)CM			00584	00584
	)DB2 PRECOMPILE			00584	00584
I -	)SET DB2PPRM1 = HOST(C),MARGINS(1,80),CCSID(1047)	RPL=	1	00585	00585
D -	)SET DB2PPRM1 = HOST(C),MARGINS(1,80)				
	)SET DB2PPRM2 = &Z	MAT=	22	00586	00586
	)SET DB2PPRM3 = &Z			00587	00587
	)CM			00588	00588
	)SET COMPPRM1 = /OPTFILE(DD:SYSOPTF)	COMPILE		00589	00589
	)SET COMPPRM2 = &Z			00590	00590
	)SET COMPPRM3 = &Z			00591	00591
	)CM			00592	00592
	)SET COMPOPT1 = NOSEQUENCE,MARGINS(1,80)	SYSOPTF DD		00593	00593
	)SET COMPOPT2 = &Z			00594	00594
	)SET COMPOPT3 = &Z			00595	00595
	*** CHANGE SECTION CUTOFF *****				

The EBCDIC CCSID for Db2 subsystem C105 is 1047, which can be viewed in the Subsystem Parameters panel DSNTIPF.



This is the DBB for SQL00002.

```

ISRBROBA NTP.S6.ACTP.STG6.\#000072.DBB(SQL00002) - 01 Line 0000000 Col 001 080
Command ==> Scroll ==> CSR
***** Top of Data *****
BIND PLAN(SQL00002) PKLIST(CMN6.SQL00002) ACT(REP) -
      ISO(CS) EXPLAIN(NO) VALIDATE(BIND) ACQUIRE(USE) RELEASE(COMMIT)
***** Bottom of Data *****

```

Stage job steps for stored procedure SQL00002 are shown in this job output.

```

      J E S 2 J O B L O G -- S Y S T E M C 0 0 1 -- N O D E M P 3 J E S 2

20.29.41 J0984224 ---- THURSDAY, 04 FEB 2016 ----
20.29.41 J0984224 IRR010I USERID SERT IS ASSIGNED TO THIS JOB.
20.29.45 J0984224 ICH70001I SERT LAST ACCESS AT 20:25:07 ON THURSDAY, FEBRUARY 4, 2016
20.29.45 J0984224 $HASP373 USER015D STARTED - INIT 1 - CLASS A - SYS C001
20.29.46 J0984224 - --TIMINGS (MINS.)-- -----PAGING COUNTS-----
20.29.46 J0984224 -STEPNAME PROCSTEP RC EXCP CONN TCB SRB CLOCK SERV WORKLOAD PAGE SWAP VIO SWAPS
20.29.46 J0984224 -SERCOPY 00 200 253 .00 .00 .0 38740 BATCH 0 0 0 0
20.29.49 J0984224 -WRITE 00 739 748 .00 .00 .0 52102 BATCH 0 0 0 0
20.29.50 J0984224 -SQL2C 00 479 226 .00 .00 .0 21822 BATCH 0 0 0 0
20.29.51 J0984224 -DB2PC 00 363 154 .00 .00 .0 15849 BATCH 0 0 0 0
20.29.52 J0984224 -BT90DBR 00 127 178 .00 .00 .0 8612 BATCH 0 0 0 0
20.30.21 J0984224 -C 00 22496 6925 .01 .00 .4 9536K BATCH 639 0 0 0
20.30.22 J0984224 -SSIDN 00 105 105 .00 .00 .0 16501 BATCH 0 0 0 0
20.30.24 J0984224 -PLKED 00 650 1115 .00 .00 .0 27544 BATCH 0 0 0 0
20.30.24 J0984224 -ALOC 00 15 11 .00 .00 .0 926 BATCH 0 0 0 0
20.30.24 J0984224 -ALOCIN 00 34 26 .00 .00 .0 3963 BATCH 0 0 0 0
20.30.26 J0984224 -LNK 00 328 195 .00 .00 .0 45995 BATCH 0 0 0 0
20.30.27 J0984224 -BT90STL 00 234 155 .00 .00 .0 38530 BATCH 0 0 0 0
20.30.28 J0984224 -CPYSTL 00 258 225 .00 .00 .0 49494 BATCH 0 0 0 0
20.30.29 J0984224 -CPYDBR 00 145 123 .00 .00 .0 18455 BATCH 0 0 0 0
20.30.31 J0984224 -SUCCESS 00 852 589 .00 .00 .0 73243 BATCH 0 0 0 0
20.30.32 J0984224 -CHKCOND 00 13 9 .00 .00 .0 955 BATCH 0 0 0 0
20.30.32 J0984224 -FAILURE FLUSH 0 0 .00 .00 .0 0 BATCH 0 0 0 0
20.30.33 J0984224 -PRINT 00 487 491 .00 .00 .0 58834 BATCH 0 0 0 0
20.30.34 J0984224 -COMPLST 00 143 171 .00 .00 .0 17747 BATCH 0 0 0 0
20.30.35 J0984224 -ILODLST 00 693 521 .00 .00 .0 45988 BATCH 0 0 0 0
20.30.37 J0984224 -USER015D ENDED. NAME-TEST TOTAL TCB CPU TIME= .03 TOTAL ELAPSED TIME= .8
20.30.37 J0984224 $HASP395 USER015D ENDED
----- JES2 JOB STATISTICS -----
04 FEB 2016 JOB EXECUTION DATE
510 CARDS READ
2,104 SYSOUT PRINT RECORDS
0 SYSOUT PUNCH RECORDS
113 SYSOUT SPOOL KBYTES
0.92 MINUTES EXECUTION TIME

```

Package promotion job steps are shown in the following job output.

```

JES2JOBLOG -- SYSTEMC001 -- NODEMP3JES2

15.22.18 J0994386 ---- TUESDAY, 09 FEB 2016 ----
15.22.18 J0994386 IRR010I USERID SERT IS ASSIGNED TO THIS JOB.
20.29.45 J0994386 ICH70001I SERT LAST ACCESS AT 15:20:23 ON TUESDAY, FEBRUARY 9, 2016
20.29.45 J0994386 $HASP373 USER015D STARTED - INIT 1 - CLASS A - SYS C001
20.29.46 J0994386 -
20.29.46 J0994386 --STEPNAME PROCSTEP RC EXCP CONN TCB SRB CLOCK SERV WORKLOAD PAGE SWAP VIO SWAPS
20.29.46 J0994386 -DDQSPQ 00 522 238 .00 .00 .0 27448 BATCH 0 0 0 0
20.29.49 J0994386 -PCLIBER FLUSH 0 0 .00 .00 .0 0 BATCH 0 0 0 0
15.22.21 J0994386 -SQLSPQ 00 208 133 .00 .00 .0 31762 BATCH 0 0 0 0
15.22.23 J0994386 -DB2PL 00 644 479 .00 .00 .0 247K BATCH 0 0 0 0
15.22.23 J0994386 -C105BND 00 218 94 .00 .00 .0 39113 BATCH 0 0 0 0
15.22.24 J0994386 -CPY1SPQ 00 59 61 .00 .00 .0 7094 BATCH 0 0 0 0
15.22.24 J0994386 -CPY1DBR 00 58 58 .00 .00 .0 7085 BATCH 0 0 0 0
15.22.25 J0994386 -CPY1LST 00 57 71 .00 .00 .0 7938 BATCH 0 0 0 0
15.22.26 J0994386 -CPY1STL 00 126 59 .00 .00 .0 14397 BATCH 0 0 0 0
15.22.26 J0994386 -CPY1PKG 00 57 56 .00 .00 .0 6823 BATCH 0 0 0 0
15.22.27 J0994386 -CPY1DBB 00 57 61 .00 .00 .0 7183 BATCH 0 0 0 0
15.22.27 J0994386 VARY WLM,APPLENV=C105SP,REFRESH
15.22.27 J0994386 -STPSTL 00 164 99 .00 .00 .0 14238 BATCH 0 0 0 0
15.22.40 J0994386 -SUCCESS 00 708 501 .00 .00 .2 52102 BATCH 0 0 0 0
15.22.40 J0994386 -CHKCOND 00 16 10 .00 .00 .0 1262 BATCH 0 0 0 0
15.22.40 J0994386 -FAILURE FLUSH 0 0 .00 .00 .0 0 BATCH 0 0 0 0
15.22.41 J0994386 -PRINT 00 97 117 .00 .00 .0 6706 BATCH 0 0 0 0
15.22.41 J0994386 -CLNLCL FLUSH 0 0 .00 .00 .0 0 BATCH 0 0 0 0
15.22.41 J0994386 -USER015B ENDED. NAME-PROMOTE TOTAL TCB CPU TIME= .01 TOTAL ELAPSED TIME= .3
15.22.41 J0994386 $HASP395 USER015B ENDED
----- JES2 JOB STATISTICS -----
09 FEB 2016 JOB EXECUTION DATE
354 CARDS READ
1,293 SYSOUT PRINT RECORDS
0 SYSOUT PUNCH RECORDS
70 SYSOUT SPOOL KBYTES
0.39 MINUTES EXECUTION TIME

```

This is the SYSPRINT output from promotion job step DDQSPQ.

```

***** TOP OF DATA *****
CMNDB2DQ: Extract SQL procedure definition from source
-----
CMNDB2DQ: Processing member SQL00002
-----
CMNDB2DQ: Processing completed max RC - 00
***** BOTTOM OF DATA *****

```

This is the SYSPRINT output from promotion job step SQLSPQ. This step interrogates Db2 subsystem C105 for an existing procedure named USER15.PROCEDURE2 and DROPS it.

The new procedure is then templated and CREATED in Db2 C105.



```

ChangeMan(R) ZMF CMNDB2DD - 8.1.0 Dynamic implementation of SQL/DDL components
CMNDB2DD      Processing begins at 15:22:20 on 02/09/2016
-----
CMNDB2DD SYSIN: TEST=YES
CMNDB2DD SYSIN: ERRSTOPAFT=0
CMNDB2DD SYSIN: DROPRC=4
CMNDB2DD SYSIN: SQLTERM=@
CMNDB2DD SYSIN: SQUEEZE=YES
CMNDB2DD SYSIN: TOLSTDNUM=YES
CMNDB2DD SYSIN: LINEFEED=NO
CMNDB2DD SYSIN: SRCSCHEMATEMPLATE=
CMNDB2DD SYSIN: TGTSCHEMATEMPLATE=
CMNDB2DD SYSIN: SRCCOLLIDTEMPLATE=
CMNDB2DD SYSIN: TGTCOLLIDTEMPLATE=
CMNDB2DD SYSIN: SRCQUALTEMPLATE=
CMNDB2DD SYSIN: TGTQUALTEMPLATE=
CMNDB2DD SYSIN: SRCOWNERTEMPLATE=
CMNDB2DD SYSIN: TGTOWNERTEMPLATE=
CMNDB2DD SYSIN: SRCWLMTEMPLATE=
CMNDB2DD SYSIN: TGTWLMTEMPLATE=
CMNDB2DD SYSIN: DB2ID=C105
CMNDB2DD SYSIN: MBR=SQL00002
SQLIN: Input cards follow ...

CREATE PROCEDURE USER15 . PROCEDURE2 ( ) RESULT SETS 1 LANGUAGE SQL EXTE
RNAL NAME SQL00002 COLLID TEST WLM ENVIRONMENT C105SP RUN OPTIONS 'TEST(
ALL,*, ,VADTCPIP&192.168.1.3:*)' P1 : BEGIN DECLARE CURSOR1 CURSOR WITH R
ETURN FOR SELECT SCHEMA , NAME FROM SYSIBM . SYSROUTINES ; OPEN CURSOR1
; END P1

CMNDD018I Statement generated by autodrop option:
DROP PROCEDURE USER15.PROCEDURE2 RESTRICT

CMNDD020I Generated drop processed successfully
CMNDD030I Work committed

CMNDD001I Templated SQL sentence extracted from member SQL00002 :

CREATE PROCEDURE USER15 . PROCEDURE2 ( ) RESULT SETS 1 LANGUAGE SQL EXTE
RNAL NAME SQL00002 COLLID TEST WLM ENVIRONMENT C105SP RUN OPTIONS 'TEST(
ALL,*, ,VADTCPIP&192.168.1.3:*)' P1 : BEGIN DECLARE CURSOR1 CURSOR WITH R
ETURN FOR SELECT SCHEMA , NAME FROM SYSIBM . SYSROUTINES ; OPEN CURSOR1
; END P1

CMNDD002I Sentence processed successfully.
CMNDD030I Work committed

-----
CMNDB2DD      Processing completed at 15:22:21 on 02/09/2016      MAX RC = 00

```

This is the SYSPRINT output from promotion job step DB2PL. No templating is used in this example.

DB2 Plan Used by ChangeMan ZMF Call Attach Facility:  
 Using plan (CMNPLAN)

Control card input (DDNAME = CMNPLCTL)

- 1 ==> TYPE=PROMOTE
- 2 ==> AUTHORITY=OWNER,INSERT
- 3 ==> INSERTQUAL
- 4 ==> USEREXIT=(ASM,NOUNLOAD)
- 5 ==> USERID=JPRESTO
- 6 ==> PACKAGE=ACTP000072
- 7 ==> PROJECT=ACTP
- 8 ==> NOBASEDBBRC=12
- 9 ==> WARNINGRC=4
- 10 ==> USEDB2PACKAGE
- 11 ==> PKLTEMPLATE
- 12 ==> DB2ID=C105
- 13 ==> LOGICAL=UNIT
- 14 ==> PLANTGT=
- 15 ==> PLANSRC=
- 16 ==> PKGETGT=
- 17 ==> PKGESRC=
- 18 ==> LOCNTGT=
- 19 ==> LOCNSRC=
- 20 ==> QUALIFIER=
- 21 ==> QUALTGT=
- 22 ==> QUALSRC=
- 23 ==> OWNER=
- 24 ==> OWNRTGT=
- 25 ==> OWNRSRC=
- 26 ==> REMOTEID=UNIT

Staged bind control statements in this change package (DDNAME = CMNPLPKG)

- 1 ==> MBR=SQL00002

Staged bind control statements in this change package (DDNAME = CMNPLDBB)

- 1 ==> MBR=SQL00002

Staged DBRMs in this change package (DDNAME = CMNPLDBR)

- 1 ==> MBR=SQL00002

The following bind control statements are required:

Templated fields which are over-long will be truncated in the following table.  
 See above (control card input) for full length templates.

Origin of Remote Reject Bind Reqmt. ID	Staged Name	Actual Name	Logical Subsys	Plan Template	Package Template	Location Template	Owner Template	Qualifier Template	DB2 Subsys
DD:PKGSSTG UNIT	SQL00002	SQL00002	UNIT		????????????????	????????????????	????????	????????	C105
DD:DBBSSTG UNIT	SQL00002	SQL00002	UNIT	????????	????????????????	????????????????	????????	????????	C105

Ending Status:

CMN7099I CMNDB2PL ending. No errors were encountered.

This is output from the promotion job DB2 BIND PACKAGE and BIND PLAN, step C105BND. No templating is used for the stored procedure walkthrough.

```

IKJ56644I NO VALID TSO USERID, DEFAULT USER ATTRIBUTES USED
READY
DSN SYSTEM(C105 )
DSN
  BIND PACKAGE(CMN6) ACTION(REP) EXPLAIN(NO) ISOLATION(CS) VALIDATE(BIND) MEMBER(SQL00002)
DSNT254I -C105 DSNTBCM2 BIND OPTIONS FOR
  PACKAGE = C105.CMN6.SQL00002.(.)
  ACTION      REPLACE
  OWNER       SERT
  QUALIFIER   SERT
  VALIDATE    BIND
  EXPLAIN     NO
  ISOLATION   CS
  RELEASE
  COPY
DSNT255I -C105 DSNTBCM2 BIND OPTIONS FOR
  PACKAGE = C105.CMN6.SQL00002.(.)
  SQLERROR    NOPACKAGE
  CURRENTDATA YES
  DEGREE      1
  DYNAMICRULES
  DEFER
  NOREOPT     VARS
  KEEPDYNAMIC NO
  IMMEDWRITE  NO
  DBPROTOCOL  DRDA
  OPTHINT
  ENCODING    EBCDIC(01047)
  PATH
DSNT232I -C105 SUCCESSFUL BIND FOR
  PACKAGE = C105.CMN6.SQL00002.(.)
DSN
  BIND PLAN(SQL00002) ACQUIRE(USE) RELEASE(COMMIT) ACTION(REP) EXPLAIN(NO) ISOLATION(CS) VALIDATE(BIND)
PKLIST(CMN6.SQL00002)
DSNT252I -C105 DSNTBCM1 BIND OPTIONS FOR PLAN SQL00002
  ACTION      REPLACE
  OWNER       SERT
  VALIDATE    BIND
  ISOLATION   CS
  ACQUIRE    USE
  RELEASE     COMMIT
  EXPLAIN     NO
  DYNAMICRULES RUN
DSNT253I -C105 DSNTBCM1 BIND OPTIONS FOR PLAN SQL00002
  NODEFER     PREPARE
  CACHESIZE   1024
  QUALIFIER   SERT
  CURRENTSERVER
  CURRENTDATA YES
  DEGREE      1
  SQLRULES    DB2
  DISCONNECT  EXPLICIT
  NOREOPT     VARS
  KEEPDYNAMIC NO
  IMMEDWRITE  NO
  DBPROTOCOL  DRDA
  OPTHINT
  ENCODING    EBCDIC(01047)
  PATH
DSNT200I -C105 BIND FOR PLAN SQL00002 SUCCESSFUL
DSN
END

```

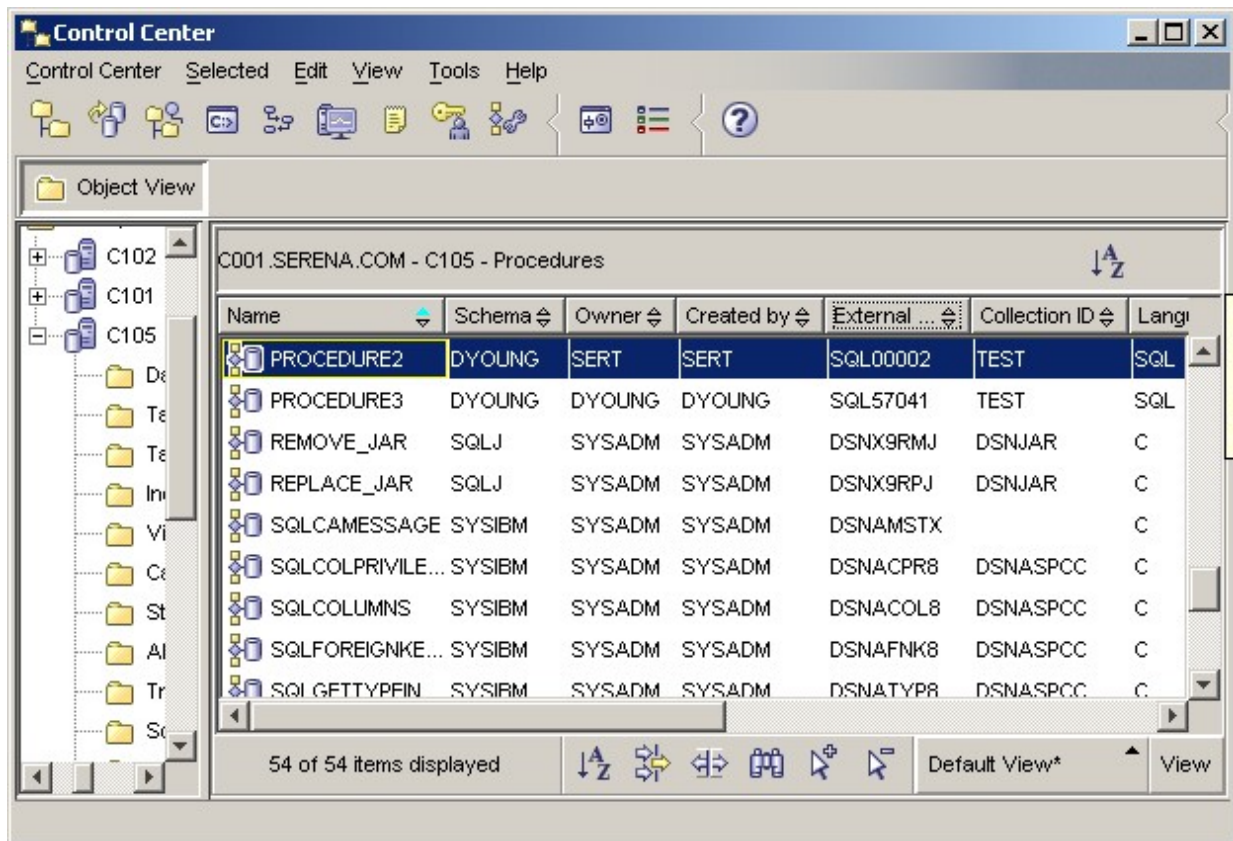
This is output from promotion job step STPSTL, which executes ZMF stored procedure utility CMNDB2SL. This utility issues z/OS commands to REFRESH the WorkLoad Manager Application ENVIRONMENT - refer to logical subsystem settings for the application.

ChangeMan(R) ZMF      CMNDB2SL - 8.1.0 Recycle Stored Procedures and Functions

CMNDB2SL Processing begins at 15:22:27 on 02/09/2016

CMNDB2SL SYSIN: TEST=NO CMNDB2SL SYSIN: DB2ID=C105  
CMNDB2SL SYSIN: WLMENVMASK=C105\*  
CMNDB2SL SYSIN: MBR=SQL00002 VARY WLM,APPLENV=C105SP,REFRESH

CMNDB2SL Processing completed at 15:22:27 on 02/09/2016 MAX RC = 00



As shown above in the Db2 Connect Control Center, SQL PROCEDURE2 was created by OWNER SERT with external name SQL00002. The same data shown in the Control Center screenshot can be retrieved using TSO/SPUFI.

# 9. Bind Service Support

---

- [Installation and Configuration](#)
- [Process Overview](#)

## Installation and Configuration

---

The support assumes that REST service versioning is active on the target Db2 subsystem. This is the default for a certain level of service on Db2 version 12. If not, the following IBM sample is used to activate it:

```
DSN1210.SDSNSAMP(DSNTIJR2)
```

The following members of the CMNZMF.CNTL distribution library have changed:

DB2OPTN

DB2OPTNR

These jobs are described in more detail below.

## DB2OPTN

---

The following columns in the CMNADMIN\_NAMED table were added in ZMF 8.2 Patch 4 to support the bind service function:

- SERVICE\_COLLECTION\_SRC IS 'Source template for COLLID - Bind service' ,
- SERVICE\_COLLECTION\_TGT IS 'Target template for COLLID - Bind service' ,
- SERVICE\_OWNER\_SRC IS 'Source template for OWNER - Bind service' ,
- SERVICE\_OWNER\_TGT IS 'Target template for OWNER - Bind service' ,
- SERVICE\_QUALIFIER\_SRC IS 'Src template for QUALIFIER - Bind service' ,
- SERVICE\_QUALIFIER\_TGT IS 'Tgt template for QUALIFIER - Bind service');

And the PROCESS\_IND column in the CMNADMIN\_GENERAL table can take a value of 'V' for Bind Service.

## DB2OPTNR

The CMNDB2VB package must now be bound at remote site Db2 subsystems – this program now performs access to Db2 catalog tables on behalf of the Bind Service process. This bind has been added:

```
BIND PACKAGE(CMNx) MEMBER(CMNDB2VB) ACT(REP) ISO(CS) -
      EXPLAIN(NO) VALIDATE(RUN) RELEASE(COMMIT) -
      ENCODING(EBCDIC) DBPROTOCOL(DRDA)
```

And the package added to the CMNPLAN pklst:

```
      BIND PLAN(CMNPLAN) -
      PKLIST(CMNx.CMNDB2SQ -
            CMNx.CMNDB2VB -
            CMNZMF.CMNDB2AT -
            *.CMNDB2SQ) -
      ACT(REP) ISO(CS) RETAIN -

EXPLAIN(NO) VALIDATE(BIND) ACQUIRE(USE) RELEASE(COMMIT)

END
```

Customers who are installing ZMF 8.2 Patch 4 and later releases from scratch can use sample JCL with these changes already in place.

Existing customers will have to make sure that they include CMNDB2VB in their remote site binds and include it in the CMNPLAN pklst (if they wish to make use of the new support).

There is a Package Master conversion involved in adding this support; but this processing is included in the standard Package Master conversion task for converting to ZMF 8.2 Patch 4 and later releases.

## Skeleton Changes

There are several skeletons addressing bind service support:

- CMN\$\$PSV local promote/demote of bind service components
- CMN\$\$RSV remote promote/demote of bind service components
- CMN\$\$BSV install/backout of bind service components.

Existing skeletons have been changed to imbed these new skeletons, for example:

- CMN\$\$PRM, CMNIMPRM imbed CMN\$\$PSV
- CMNRPMCR, CMNRPICR imbed CMN\$\$RSV
- CMN21, CMN49 imbed CMN\$\$BSV

# ZMF Global Administration Changes

## Create the Library Type Definitions

If the libtype is PDS/E based, use an LRECL of 80. In the entries that follow the libtypes used mostly for testing are BSZ for bind service and BSG for grants.

All library types must be given the D (Db2) selectable option Panel CMNCGLT0.

```

CMNCGLT0          Global Library Types Part 1 of 2          Row 211 to 213 of 213
Command ==>> _____ Scroll ==>> CSR

   Lib              Order  Lke  Seq  Defer  Target  Sel
   type Description      +                type  Opt
____ BSP Bind Service PDSE based components    0  P  ___  Y    ___  D
____ BSZ Bind Service zFS based components     0  P  ___  Y    ___  D
____ BSG Bind Service Grant components         0  P  ___  Y    ___  D
***** Bottom of data *****
  
```

In the ZMF Db2 option administration, each of these library types needs a Db2 subtype of 'V' panel CMNDB2UP:

```

CMNDB2UP          Db2 Library Types          Row 15 to 17 of 17
Command ==>> _____ Scroll ==>> CSR

   Lib              Db2      End SQL
   type Description      sub      sentence
   BSP Bind Service PDSE based components      V      ___
   BSZ Bind Service zFS based components      V      ___
   BSG Bind Service Grant components          V      ___
***** Bottom of data *****
  
```

You need to define all the logical subsystems that your applications may use (this is normal). Panel CMNGD2LN - No change here apart from a new set of templates specifically for Bind Service:

```

CMNGD2LN                               Db2 Logical Subsystems

Command ==> _____ Scroll ==> CSR

Line commands:
P Specify miscellaneous parameters
T B Bind plan/pkg process named(T) and general(B) templates
Q G SQL process          named(Q) and general(G) templates
V H Bind service process named(V) and general(H) templates

      Logical   Db2
      name      subsys   Site      Description
____ BASELINE  D10L    U900DP  U900DP BASELINE
____ PRODLCL1  D10L    U900DP  U820DP PRODUCTION \#1
____ PRODLCL2  D10L    U900DP  U820DP PRODUCTION \#2
____ PRODRMT1  D10L    U900P   U820P  PRODUCTION \#1
____ PRODRMT2  D10L    U900P   U820P  PRODUCTION \#2
____ SYSTEST1  D10L    STEVEPRM REMOTE PROMOTION FOR STEV
____ UNIT1     D10L    STEVEPRM REMOTE PROMOTION FOR STEV
***** Bottom of data *****

```

If you use line action P against a logical subsystem, you can set the library type that will be used to process bind service grants – this is a new field on panel CMNGD2PM:

```

CMNGD2PM                               Db2 Logical Subsystem PRODRMT1 Settings

Command ==> _____

Preferred Libtypes:
DBRM . . . . . DBR
Plan bind parameters . . . . . DBB
Package bind parameters . . . . . PKG
Service grants . . . . . BSG

General Parameters:

Enter "/" to select option
 / Bind Failure is significant
 / Recycle Stored Procedures where WLM Environment is . . _____
 / Maintain Trigger Sequence
 / Use Db2 versioning for Native SQL Stored Procedures

```

**Use Db2 Versioning for Native SQL Stored Procedures**

Using V allows you to specify model templates (specific values are left to the application definitions). For example panel CMNGD2L6:



```

CMNGD2L6      Db2 Logical Subsystem PRODRMT1 Bind Service Named Templates
Command ==> _____

Templates      Target          Source

Collection . . _____ + _____ +
Qualifier . . _____ + _____ +
Owner . . . . _____ + _____ +

```

And H the general token templates panel CMNGD2L7:

```

CMNGD2L7 Db2 Logical Subsystem PRODRMT1 Bind Service General Row 1 to 13 of 13
Command ==> _____ Scroll ==> CSR

Token name      + Target template      + Source template      +
_____
_____

```

as stated earlier, values are usually left to the application level administration.

## ZMF Application Administration Changes

Again, using a sample application named STEV, we have the library types panel CMNCLLT0:

```

CMNCLLT0      STEV - Library Types Part 1 of 2      Row 211 to 213 of 213
Command ==> _____ Scroll ==> CSR

Lib            Order  Lke  Seq  Defer  Target  Sel
type Description  +      type  Opt
_____ BSG Bind Service Grant components      0  P  ___  Y  ___  D
_____ BSG Bind Service PDSE based components 0  P  ___  Y  ___  D
_____ BSZ Bind Service zFS based components 0  P  ___  Y  ___  D
***** Bottom of data *****

```

This time we need to define their baseline repositories panel CMNCBAS1:

```

CMNCBAS1      STEV - Baseline Configuration Part 1 of 2      Row 5 to 24 of 51
Command ==> _____ Scroll ==> CSR

Type  Levels  Install  Baseline
      in prod  storage
_____
BSG   2       Y       H
BSP   10      Y       SD
BSZ   2       Y       H

```

This should be followed either by an I/A extract and reload or just use XML, for example:

```

<?xml version="1.0"?>
<service name="IMPACT">
  <scope name="BUN">
    <message name="CREATE">
      <header>
        <subsys>M</subsys>
        <product>CMN</product>
      </header>
      <request>
        <appl>STEV</appl>
        <libType>BSZ</libType>
        <libLike>P</libLike>
        <baseline>/cmndev/STEV/Base00/BSZ</baseline>
      </request>
    </message>
  </scope>
</service>

```

And so on.

You need to provide some promotion definitions, for example panel CMNLRPM3:

CMNLRPM3		STEV/STEVPRM - Promotion Libraries		Row 2 to 6 of 23	
Command ==>				Scroll ==> CSR	
Promotion name: UNIT1		Level: 10			
Lib	Syslib exclude	Cleanup Level	Target libraries		
_____ BSG	Y	Y	/cmndev/STEV/promo10/BSG /cmndev/STEV/promo10/BSG	+ Shadow	
				+ Library 1	
				+ Library 2	
				+ Library 3	
_____ BSP	Y	Y	WSER58.PROMO10.BSP WSER58.PROMO10.BSP	+ Shadow	
				+ Library 1	
				+ Library 2	
				+ Library 3	
_____ BSZ	Y	Y	/cmndev/STEV/promo10/BSZ /cmndev/STEV/promo10/BSZ	+ Shadow	
				+ Library 1	
				+ Library 2	
				+ Library 3	

And some production libraries/directories panel CMNCPRDL:

CMNCPRDL STEV - U900P Production Libraries Row 1 to 6 of 6  
 Command ==> \_\_\_\_\_ Scroll ==> CSR

```

Type Production dataset name +
Temporary dataset name +
Backup dataset name +
_____ BSG /cmndev/STEV/U900P/Prod00/BSG
          /cmndev/STEV/Temp/BSG
          /cmndev/STEV/U900P/Prod01/BSG
_____ BSP WSER58.PROD.BSP
          WSER58.TEMP.BSP
          WSER58.PROD.BSP.BKUP
_____ BSZ /cmndev/STEV/U900P/Prod00/BSZ
          /cmndev/STEV/Temp/BSZ
          /cmndev/STEV/U900P/Prod01/BSZ
  
```

Then, in the Db2 option definitions, you need to identify the new libtypes with subtype V panel CMNDB2UP:

CMNDB2UP STEV - Db2 Library Types Row 8 to 10 of 10  
 Command ==> \_\_\_\_\_ Scroll ==> CSR

Lib type	Description	Sub type	End SQL sentence
BSP	Bind Service PDSE based components	V	—
BSZ	Bind Service zFS based components	V	—
BSG	Bind Service Grant components	V	—

In the Db2 active library definitions you associate a target library/directory with a Db2 logical subsystem. For example panel CMNLD2AL:

CMNLD2AL Db2 Active Library/Directory List Row 6 to 26 of 35  
 Command ==> \_\_\_\_\_ Scroll ==> CSR

Logical name	Type	Db2 active library or directory name	
PRODLCL1	V	/cmndev/STEV/U900DP/Prod00/BSG	+
PRODLCL1	V	/cmndev/STEV/U900DP/Prod00/BSZ	+
PRODRMT1	V	/cmndev/STEV/U900P/Prod00/BSZ	+
PRODRMT1	V	/cmndev/STEV/U900P/Prod00/BSG	+
UNIT1	V	/cmndev/STEV/promo10/BSZ	+
UNIT1	V	/cmndev/STEV/promo10/BSG	+

Now we consider the zFS libtypes BSZ and BSG. You have to set up the logical subsystems to do what you want them to do. For example panel CMNGD2LN:

```

CMNGD2LN                Db2 Logical Subsystems                Row 1 to 6 of 6
Command ==> _____ Scroll ==> CSR

```

Line commands:

- P Specify miscellaneous parameters
- T B Bind plan/pkg process named(T) and general(B) templates
- Q G SQL process named(Q) and general(G) templates
- V H Bind service process named(V) and general(H) templates

	Logical name	Db2 subsys	Site	Description
_____	PRODLCL1	D10L	U900DP	U900DP PRODUCTION \#1
_____	PRODLCL2	D10L	U900DP	U900DP PRODUCTION \#2
_____	PRODRMT1	D10L	U900P	U900P PRODUCTION \#1
_____	PRODRMT2	D10L	U900P	U900P PRODUCTION \#2
_____	SYSTEST1	D10L	STEVEPRM	REMOTE PROMOTION FOR STEV
_____	UNIT1	D10L	STEVEPRM	REMOTE PROMOTION FOR STEV

\*\*\*\*\* Bottom of data \*\*\*\*\*

Using UNIT1 as an example:

Use P to set the grant libtype panel CMNGD2PM:

```

CMNGD2PM                Db2 Logical Subsystem UNIT1 Settings
Command ==> _____

Preferred Libtypes:
DBRM . . . . . DBR
Plan bind parameters . . . . . DBB
Package bind parameters . . . . . PKG
Service grants . . . . . BSG

General Parameters:
Enter "/" to select option
 / Bind Failure is significant
 _ Recycle Stored Procedures where WLM Environment is . . _____
 _ Maintain Trigger Sequence
 _ Use Db2 versioning for Native SQL Stored Procedures

```

Use V to set the named templates panel CMNGD2L6:

```

CMNGD2L6                Db2 Logical Subsystem UNIT1 Bind Service Named Templates
Command ==> _____

Templates                Target                Source

Collection . . . . . DEVU                + PRD                +
Qualifier . . . . . WSER58-----+ _____        +
Owner . . . . . SERD-----+ _____        +

```

And, use H to set the general token templates panel CMNGD2L5:

```

CMNGD2L5   Db2 Logical Subsystem UNIT1 BIND General Template   Row 1 to 3 of 3
Command ==> _____ Scroll ==> CSR

      Token name          + Target template          + Source template          +
_____ >GRANTEE<          DBCORP4          'DBCORPR'
_____ >GRANTEE<          TGENID1          'GENID1'
_____ >GRANTEE<          TGENID3          'GENID2'
***** Bottom of data *****

```

Here we see the special general token template that will only be used for authorization ids in a bind service grant list. From the ISPF help panel CMN94033:

There is a special form of general token templates implemented for this function. This is only applied to any grant SQL components associated with the bind service process.

If the token name is >GRANTEE< then the template will be applied to the list of grantee userids/groups on the grant SQL supplied to the process.

An extension to the search and replace algorithm is available (for this function only). If you put the source template in single quotes then the target template will only be used if the value to be replaced exactly matches the source template value.

As an example of standard search and replace templating, against this list of grantees:

```
PRD, PRD1, ISOL8, APRD:
```

General Token name: >GRANTEE< source template: PRD target template: DEV results in: DEV, DEV1, ISOL8, ADEV.

As an example of 1-to-1 search and replace templating:

General Token name: >GRANTEE< source template: 'PRD' target template: DEV results in: DEV, PRD1, ISOL8, APRD.

## Process Overview

Bind service components come in two flavors:

There is the bind service component itself, which consists of the bind service command and any related service SQL. The bind command is placed first in the component and terminated with a semicolon. The SQL follows on from the semicolon (see below for examples).

There is the service package grant component, which specifies the userids/groups who are allowed to execute the service package.

Both sets of components have templates applied to them so that the production version of the components can be modified to be applied to the various test levels through the lifecycle.

These components can be hosted by PDS/E library types or by zFS library types. The latter are more useful as the component (both flavors) must have the same name as the service itself. Db2 allows the service name to be up to 128 bytes in length (a PDS/E libtype would restrict this to 8 bytes).

An example of a bind service component in a package staging directory is:

```
VIEW      /cmndev/cmnj/STEV/#000445/BSZ/Get_Area_Regression_Info
Command ==> _____
***** Top of Data *****
000001 BIND SERVICE(PRD) +
000002 NAME("Get_Area_Regression_Info") +
000003 SLENCODING(EBCDIC) +
000004 DESCRIPTION('CREATED USING BIND SERVICE') +
000005 PATH( PRD , SCHEMA1 , +
000006 SCHEMA2, +
000007 SCHEMA3 ) +
000008 QUALIFIER(CMNJ) +
000009 ISOLATION(UR) +
000010 OWNER(WSER58) ;
000011 CALL GETAREA_REGINFO (:AREANAME)
***** Bottom of Data *****
```

The component name is **Get\_Area\_Regression\_Info**, which is the same as the service name.

The bind service command is terminated by the semicolon on line 10. The service SQL is on line 11.

The bind service command will result in a Db2 package being created which will have a name the same as the NAME clause and a collection id the same as the SERVICE clause.

Templates are provided to allow the ZMF processes (promote, and so on) to change the collection id, the qualifier, and the owner.

General token templates will also be provided for anything else. Only the bind service command itself is templated; the SQL is not touched.

Clauses such as the qualifier will be applied by Db2 to the SQL it executes on behalf of the service call.

For now, each bind parameter and SQL line must be completed within the first 72 columns of the component (just like a standard bind command).

This is an example of the bind service grant component associated with this bind service component (that is, it has the same name, different libtype):

```

VIEW      /cmndev/cmnj/STEV/#000445/BSG/Get_Area_Regression_Info
Command ==>
***** Top of Data *****
000001 GRANT EXECUTE ON PACKAGE PRD."Get_Area_Regression_Info" TO DBCORPR, GENID1,
000002 GENID2, GENID12;
***** Bottom of Data *****

```

You can have as many GRANTS as you like in this component, separated by semicolons. The authorization id's can be listed all on one GRANT or separated onto individual GRANTS or a mixture.

Once again, templating will be applied to the collection id (PRD in the above example). There is a special general token template (>GRANTEE<) which can be used against the authorization id list (see below).

The GRANT SQL must be contained within the first 80 bytes of each record.

## Batch Utility Overviews

### CMNDB2SV – Process Bind Service Parameter/SQL and Grant Request Components

This program takes the bind service component and splits it up into the bind command and the associated SQL for the service. It also processes any related grant request components.

Bind parameters and grant requests are templated as described above. There are (potentially) three resultant parameter sets output for each named service:

1. The bind command for the service
2. The SQL associated with the service
3. The grant and/or revoke requests associated with the service.

The bind parameters and the SQL are passed to a subsequent execution of IKJEFT01, which will run the binds. The grant requests are passed to a subsequent execution of CMNDB2GR (details below).

Here is some typical JCL showing the promotion of the components associated with two Db2 REST services:

```

//SRVBSZ EXEC PGM=CMNDB2SV,      *** BIND SERVICE PROCESS
//          COND=(4,LT),        *** FOR LIBTYPE BSZ
//          REGION=0M
//SYSPRINT DD  SYSOUT=*
//SERPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//CMNZFSIN DD  PATH='/cmndev/CMNKP/STEV/#000436/BSZ',
//          PATHOPTS=ORDONLY
//CMNGRANT DD  PATH='/cmndev/CMNKP/STEV/#000436/BSG',
//          PATHOPTS=ORDONLY
//*
//SYSIN      DD  *
TYPE=PROMOTE
DB2ID=D10L
LOGICAL=UNIT1
GENERATEREVOKES=YES
COLLSRC=PRD
COLLTGT=DEVU
QUALSRC=
QUALTGT=WSER58-----
OWNRSRC=
OWNRTGT=SERD-----
TOKNAME=>GRANTEE<
TOKNSRC='DBCORPR'
TOKNTGT=DBCORP4
TOKNAME=>GRANTEE<
TOKNSRC='GENID1'
TOKNTGT=TGENID1
TOKNAME=>GRANTEE<
TOKNSRC='GENID2'
TOKNTGT=TGENID3
FILE=SQL00001,zFScomponentName001
FILE=SQL00002,zFScomponentName002
/*
/**
/** OUTPUT TEMPORARY FILES FOR THE BIND SERVICE COMPONENT SQL
/**
//SQL00001 DD DISP=(,PASS),DSN=&&SQL00001,
//          UNIT=SYSDA,SPACE=(TRK,(5,10),RLSE),
//          DCB=(DSORG=PS,LRECL=80,RECFM=FB,BLKSIZE=0)
//SQL00002 DD DISP=(,PASS),DSN=&&SQL00002,
//          UNIT=SYSDA,SPACE=(TRK,(5,10),RLSE),
//          DCB=(DSORG=PS,LRECL=80,RECFM=FB,BLKSIZE=0)
/**
/** OUTPUT TEMPORARY FILE FOR THE BIND SERVICE PARAMETERS
/**
//D10LBCTL DD DISP=(MOD,PASS),DSN=&&D10LBCTL,
//          UNIT=SYSDA,SPACE=(TRK,(15,1),RLSE),
//          DCB=(DSORG=PS,LRECL=80,RECFM=FB,BLKSIZE=0)
/**
/** OUTPUT TEMPORARY FILE FOR THE GRANT SQL
/**
//D10LGCTL DD DISP=(MOD,PASS),DSN=&&D10LGCTL,
//          UNIT=SYSDA,SPACE=(TRK,(15,1),RLSE),
//          DCB=(DSORG=PS,LRECL=80,RECFM=FB,BLKSIZE=0)

```

The skeletons are set up to cope with bind service and grant requests components at the same time, or individually.



The output DDnames for the service SQL are generated dynamically by skeleton logic – these same DDnames must be used by the follow-on IKJEFT01 step.

One of the bind service parameters is SQLDDNAME, which is used by the bind service command to find the SQL associated with the service. This DDname, if specified in the input component, is ignored (and, in fact, stripped out); we generate our own SQLDDNAME parameter value for each service being processed.

The input bind service components are read from the CMNZFSIN ddname (if the staging library is a zFS libtype; CMNPDSIN if not). The separator between the bind parms and the associated SQL is a semicolon. For example:

```
BIND SERVICE(PRD) -
  NAME("zFScomponentName001") -
  SQLDDNAME(SQL) -
  SQUELCOING(EBCDIC) -
  DESCRIPTION('From an ERO package') -
  QUALIFIER(SCD) -
  ISOLATION(UR) -
  OWNER(WSER58) ;
SELECT * FROM CMNADMIN_GENERAL
  WHERE PROCESS_IND = 'V'
```

Note that the SQLDDNAME parm is included above. CMNDB2SV will remove this before adding our own. (In this example we will replace it with SQLDDNAME(SQL000001).)

After separating the two parts of the component, CMNDB2SV will apply templates as specified in the sysin. (See below for full list of sysin parms for CMNDB2SV).

It will restructure the bind parameters to make applying templates more tractable. In this example, the bind parms written to D10LBCTL, after templates have been applied, look like this:

```
BIND SERVICE(DEVU) +
  NAME("zFScomponentName001") +
  SQUELCOING(EBCDIC) +
  DESCRIPTION('From an ERO package') +
  QUALIFIER(WSER58) +
  ISOLATION(UR) +
  OWNER(SERD) +
  SQLDDNAME(SQL00001)
```

If Grant request components are present, these components are also restructured. They are broken up into one grant request per service and authorization id and templates applied (especially note the >GRANTEE< special template described above).

For example, the grant request component for this particular service looks like this:

```
GRANT EXECUTE ON PACKAGE PRD."zFScomponentName001" TO DBCORPR,
GENID1, EROID, GENID3, BILYBOB, GENID12;
```

Note that grant request parms must be contained within the first 80 bytes of the record.

After templating CMNDB2SV, put the following out to DDname D10LGCTL:

```
GRANT EXECUTE ON PACKAGE DEVU."zFScomponentName001" TO DBCORP4;
GRANT EXECUTE ON PACKAGE DEVU."zFScomponentName001" TO TGENID1;
GRANT EXECUTE ON PACKAGE DEVU."zFScomponentName001" TO EROID;
GRANT EXECUTE ON PACKAGE DEVU."zFScomponentName001" TO GENID3;
GRANT EXECUTE ON PACKAGE DEVU."zFScomponentName001" TO BILYBOB;
GRANT EXECUTE ON PACKAGE DEVU."zFScomponentName001" TO GENID12;
```

CMNDB2SV will query the Db2 catalog to see whether a grant request is actually needed. If the request is already satisfied, it will not be passed on (to avoid unnecessary SQL errors stating that the grantee already has the privilege, sqlcode +562). Requests to grant privileges to the current authorization id (that is, the RACF userid of the promotion (and so on) job being run) will not be propagated (to avoid -554 sql errors).

Similarly, if GENERATEREVOKE=YES is used, and if catalog grantee's exist that are not reflected in the grant component for this service, then REVOKE requests will be generated to have them removed from the catalog.

Revoke requests for the current authorization id will not be generated (that is, they would lead to -555 sql errors).

The skeletons are set up to cope with just bind service components, just grant components, or both in any one run.

SYSIN parameters for CMNDB2SV (records that start with an asterisk are comments and are ignored):

<b>SYSIN Parameter</b>	<b>Description</b>
TYPE=[PROMOTE/ DEMOTE/ INSTALL/ BACKOUT	Processing differs slightly for each type of ZMF lifecycle action. There is no default.
TRACE=YES/NO	Db2 CAF traffic will be traced to SERPRINT (default is NO).
AUTOFREE=YES/NO	If the service to be bound is already present at the target Db2, do we issue a FREE SERVICE first? Note that there is no such thing as ACTION(REPLACE) for a service bind. If the service is already present, the bind will fail. The default for the AUTOFREE parm is YES.


<b>SYSIN Parameter</b>	<b>Description</b>
GENERATEREVOKES=YES/ NO	When processing grant requests, do we want to generate REVOKE requests for ids that currently have authority for the target bind service but that do not appear in the (templated) list presented in the grant component? For example, if userid WSER58 has execute authority on DEVU."zFScomponentName001" will we generate a REVOKE for it based on the fact that it doesn't appear in the list presented above? The default for the GENERATEREVOKES parm is NO. If this parm is set to YES then, in the example above, CMNDB2SV would generate this REVOKE request: REVOKE EXECUTE ON PACKAGE DEVU."zFScomponentName001" FROM WSER58 NOT INCLUDING DEPENDENT PRIVILEGES; DB2ID=ssss where ssss is the target Db2 subsystem id for this set of processes. There is no default.
LOGICAL=*xxxxxxxx	The ZMF logical subsystem name that has generated the templates and so on for this run. This name is only used in messaging; CMNDB2SV makes no actual connection to ZMF.
COLLSRC=	The source template for the collection id. The collection id is the value specified in the BIND SERVICE() parameter and is used by Db2 to qualify the package name for the service.
COLLTGT=	The target template for the collection id.
QUALSRC=	The source template for the qualifier.
QUALTGT=	The target template for the qualifier.
OWNRSRC=	The source template for the owner.
OWNRTGT=	The target template for the owner.
TOKNAME=	The name of a general token (>GRANTEE< is a special case, see above).
TOKNSRC=	The source template for this token.
TOKNTGT=	The target template for this token.

You can specify as many TOKNAME, TOKNSRC, TOKNTGT general token triplets as you need.

SYSIN Parameter	Description
FILE= <i>sqlddname</i> , *ffffffffffffffffffffffffffffffff	zFS file name to be read from the CMNZFSIN DD statement to be parsed for DDL/SQL to be templated and passed through to the <i>ddddBCTL</i> ddname.
MBR= <i>sqlddname</i> , *mmmmmmmm	MVS member name to be read from the CMNPDSIN DD statement to be parsed for DDL/SQL to be templated and passed through to the <i>ddddBCTL</i> ddname. For both FILE= and MBR=, the first parameter, <i>sqlddname</i> , is intended to be generated by the file tailoring process. This name will be inserted into the SQLDDNAME= bind service parameter and will be added to the follow-on bind step.

The case of the SYSIN keywords is not significant. Also, where the keyword value is unambiguous (for example, YES/NO, PROMOTE/INSTALL/and so on) the specified value is not case-sensitive.

CMNDB2SV will generate the SQL for this service into a temporary dataset which will be passed to the bind step.

 **Note**

For isolation testing purposes you can replace the output ddnames in the CMNDB2SV step with SYSOUT to see what the program has generated without actually doing anything.

CMNDB2SV itself does not change anything in the Db2 catalog. For example, in the above step you could specify this:

```

/**
/** OUTPUT TEMPORARY FILES FOR THE BIND SERVICE COMPONENT SQL
/**
//SQL00001 DD SYSOUT=*
//SQL00002 DD SYSOUT=*
/**
/** OUTPUT TEMPORARY FILE FOR THE BIND SERVICE PARAMETERS
/**
//D10LBCTL DD SYSOUT=*
/**
/** OUTPUT TEMPORARY FILE FOR THE GRANT SQL
/**
//D10LGCTL DD SYSOUT=*

```

## CMNDB2GR - Process Grant Requests Passed by CMNDB2SV

Note that we could equally well process the grant requests using the IBM-supplied sample program DSNTDP2. However, some value is added with the CMNDB2GR utility.

The program reads the grant/revoke requests passed by CMNDB2SV (or anything else for that matter) and presents them to the target Db2 and reports back on the success of the operation.

It allows you to create a list of SQL codes which are of no significance as well as to set a return code of your choosing should any of the requests fail. Typical JCL, taken from a promote job, looks like this:

```

//D10LGRN EXEC PGM=CMNDB2GR, *** PERFORM GRANTS
// COND=(4,LT)
//SYSPRINT DD SYSOUT=*
//SERPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
/**
/** GRANT SQL
/**
//CMNGRANT DD DISP=(OLD,DELETE),DSN=&&D10LGCTL
//SYSIN DD *
db2Id=D10L
ignoreSQLcode=-556
/*

```

The available SYSIN parameters are (records that start with an asterisk are comments and are ignored):

<b>SYSIN Parameter</b>	<b>Description</b>
TRACE=YES/NO	Db2 CAF traffic will be traced to SERPRINT. Default is NO.
ERRORRC=*nnn	Set up to three-digit step completion code should any of the requests be deemed to have failed. Default is 8.
DB2ID=*ssss	The target Db2 subsystem id for this set of processes. No default.

<b>SYSIN Parameter</b>	<b>Description</b>
IGNORESQLCODE=-nnnnn or +*nnnnn	An SQL code that is not to be considered as an error situation. That is, the step condition code will remain zero if this sqlcode is encountered for one or more requests. You can code up to 1,024 of these parameters.
PROMOTE	The promote action will attempt to bind the Db2 REST service requests and process any grant components included in the promotion.
DEMOTE	The demote action will free the Db2 REST services. No grant/revoke processing will take place. Freeing a service will result in authorities granted to that service being dropped from the catalog anyway.
INSTALL	The install action will attempt to bind the Db2 REST service requests and process any grant components included in the install.
BACKOUT	The backout action will attempt to bind the Db2 REST service requests and process any grant components included in the original install but based on the components as they exist in the backout library (if they exist).

The case of the SYSIN keywords is not significant. Also, where the keyword value is unambiguous (for example, YES/NO, PROMOTE/INSTALL/and so on) the specified value is not case-sensitive.

# 10. Db2 Option User Exits

---

This appendix describes the exit programs used in the ChangeMan ZMF Db2 Option.

- CMNEX101 Bind Control Statement Processor
- CMNEX103 Bind Control Statement Triage
- CMNDB2DD - HLL exit

## CMNEX101 Bind Control Statement Processor

---

Exit program CMNEX101 lets you modify BIND PLAN and BIND PACKAGE commands beyond what is allowed by logical subsystem templating in the ChangeMan ZMF Db2 Option. You can create BIND commands that are not found in staging and baseline libraries.

Plan lookup program CMNDB2PL can call the user exit program CMNEX101 twice during the processing of a BIND command record set:

1. CMNEX101 is called if no BIND command members are found for a staged DBRM and if keyword CREATEECC is input to CMNDB2PL. User code in the exit can create the missing BIND command, which is passed back to CMNDB2PL and written to the output ddname `stssysBCTL`.
2. CMNEX101 is called before each templated BIND command record set is written to ddname `stssysBCTL`. The exit can examine the BIND command records passed from CMNDB2PL and:
  - a. Accept the BIND command records and let CMNDB2PL write the records to the output BIND command file as they are.
  - b. Modify or replace the BIND command records and pass them back to CMNDB2PL for output.
  - c. Reject the BIND command set, tell CMNDB2PL to exclude the records from the output BIND command file, but let CMNDB2PL to continue processing the next BIND command.
  - d. Reject the BIND command set, tell CMNDB2PL to exclude the records from the output BIND command file, let CMNDB2PL to continue processing the next BIND command, but have CMNDB2PL terminate with a return code that will stop job processing.
  - e. Reject the BIND command set and tell CMNDB2PL terminate immediately with a return code that will stop job processing.

## Important

Keyword operands in a BIND command that CMNDB2PL sends to exit program CMNEX101 may be in a different order than in the original BIND command in a staging, promotion, baseline, or production library. See [BIND Command Keyword Option Order](#).

## CMNDB2PL Parameter Passed to CMNEX101

CMNDB2PL passes data to exit program CMNEX101 in the communication area defined by copybook CMNEX101, which is delivered in the CMNZMF ASMCPY library.

See copybook CMNEX101 for a list of the fields passed to CMNEX101 and a description of each field or indicator.

## Activating CMNEX101

Activate exit program CMNEX101 by commenting out the line of code displayed below:

```
CMNEX101 RMODE ANY
* Comment (or delete) the following 1 line to activate this exit.
      DC Y(2046)                inactive module
```

## CMNEX101 Process

See the program comments for a description of CMNEX101 processing, particularly the comments under these headings:

- Accessing PKLIST elements
- Accessing DBRM list elements

## CMNEX101 Usage Scenarios

As the following three scenarios illustrate, three types of CMNDB2PL requests are passed to CMNEX101. These request types reside within the CMNEX101 copybook and are represented by the X101\$IND indicator and X101CNT field.

### Scenario 1

- Input Request: TM X101\$IND,X101\$CC



- Create bind control statements for members not found in the specified staging or baseline libraries.
- X101WORK is provided to CMNEX101 as a work area to build the necessary bind control statements.
- Output Request: OI X101\$IND,X101\$CC
  - RETCODE field must be set to a value of '4'.
  - The X101WORK work area must contain the bind control statements created for the members not found in the processed staging or baseline libraries.
  - X101CNT is required to contain the number of bind control statements placed in the work area represented by X101WORK.
- Example of a Create Request:
  - Activate CMNEX101. See [Activating CMNEX101](#).
  - Assemble and link edit using the reentrant parameter.
  - Create a DBB member that does not exist in the target DD names DBBSSTG and DBBSBAS.
  - Specify SYSIN parameter CREATCC.
  - Invoke CMNDB2PL with this DBB member name.
  - The \_\_\_BCTL DD output will contain an example of bind control cards that have been created for the DBB member.

## Scenario 2

- Input Request: TM X101\$IND,X101\$CHK
  - A set of bind control statements are passed to CMNEX101.
  - X101BIND work area contains the set of bind control statements.
  - X101CNT contains the number of bind control statements passed in the X101BIND work area.
- Output Request A: OI X101\$IND,X101\$REP
  - RETCODE field must be set to a value of '4'.
  - X101WORK work area must contain the set of bind control statements that will replace the set of bind control statements represented in the X101BIND work area.
  - X101CNT is required to contain the number of bind control statements passed in the X101WORK work area.
- Example of a Replace Request:

- Activate CMNEX101. See [Activating CMNEX101](#).
  - Assemble and link edit using the reentrant parameter.
  - Create a DBB member named TSTMBR2 in the staging library.
  - Invoke CMNDB2PL with this DBB member name.
  - The \_\_\_BCTL DD output will contain an example of bind control cards that have been replaced the original contents of the TSTMBR2 member.
- Output Request B: OI X101\$IND,X101\$ADD
    - RETCODE field must be set to a value of '4'.
    - 101WORK work area must contain the bind control statements that will be added (appended) to the bind control statement set represented in the X101BIND word area.
    - X101CNT is required to contain the number of bind control statements passed in the X101WORK work area.
  - Example of an Add Request:
    - Activate CMNEX101. See [Activating CMNEX101](#).
    - Assemble and link edit using the reentrant parameter.
    - Create a DBB member named TSTMBR1 in the staging library.
    - Invoke CMNDB2PL with this DBB member name.
    - The \_\_\_BCTL DD output will contain an example of bind control cards that have been added the original contents of the TSTMBR1 member.

### Scenario 3

- Input Request: X101CNT equal to =F'1'
- This is a request for the set of bind control statements that was previously passed to CMNEX101. If needed, CMNDB2PL allows CMNEX101 to perform a cleanup process.

## CMNEX101 Return Codes

Before returning to CMNDB2PL, CMNEX101 will issue one of the following return codes (in register 15).

Code	Explanation
00	The BIND command records passed by CMNDB2PL are acceptable and to be used as is.

Code	Explanation
04	CMNEX101 made changes to the BIND command records and placed them in the third work area. The number of records in the third work area is indicated in the communications area. CMNDB2PL will write the modified records to ddname <i>stssysBCTL</i> .
08	Indicates that CMNEX101 rejects the passed BIND command record set. CMNDB2PL will not write this set of BIND command records to the output ddname <i>stssysBCTL</i> , but it will continue processing with the next set of BIND command records. CMNDB2PL will issue the warning return code.
12	CMNEX101 rejects the passed BIND command record set, but CMNDB2PL is to continue processing with the next set of BIND command records. CMNDB2PL will terminate with RC=12 after processing all BIND command sets.
16	Indicates that CMNEX101 rejects the passed BIND command record set, and CMNDB2PL will terminate immediately with RC=12.

## CMNEX103 Bind Control Statement Triage

Program CMNDB2PL calls exit CMNEX103 and passes BIND statement images immediately prior to parsing them for keywords.

CMNEX103 strips records you designate out of the BIND command set and writes them to a ChangeMan ZMF temporary data set that is allocated by CMNDB2PL. Retained records are later appended to the end of the BIND command set after templating and before exit CMNEX101 is called.

BIND commands that are input to CMNDB2PL are parsed with IBM service routine IKJPARS to ensure that CMNDB2PL processing is synchronized with IBM changes to BIND keyword operands. IKJPARS does not pass along "comment" records with asterisk (\*) in position 1, which some user sites use to control the behavior of exit program CMNEX101.

Sample code in exit program CMNEX103 shows you how to preserve comment records that are input to CMNDB2PL.

### Important

Keyword operands in a BIND command that CMNDB2PL sends to exit program CMNEX103 may be in a different order than in the original BIND command member in a staging, promotion, baseline, or production library. See [BIND Command Keyword Option Order](#).

## CMNDB2DD - HLL exit

---

This HLL exit, while developed with Native SQL SPs in mind, can apply equally well to any DDL/SQL object which is processed by CMNDB2DD.

As mentioned in the description of the new CMNDB2DD sysin parameter 'HLLX=', CMNDB2DD now takes a High Level Language user exit. However, note that the mechanism for this is not via the standard HLLX stc call process. CMNDB2DD calls the exit directly itself and only makes use of the existing HLLX 'paradigm' in that it can call either a REXX exec or an LE-enabled language load module. If the exit is a REXX exec then the JCL for the CMNDB2DD step must include a // HLLXREXX dd statement from which the exec will be loaded. If it is an LE-program then it will be loaded from the usual places (step/joblib).

The SQL sentence currently being processed is written to a temporary file with the ddname of DDLIN, the exit can work with that ddname, read the DDL from the file, validate or change it. If validation fails then the relevant 'proceed' variable can be set to 'NO' and a message can be sent back to CMNDB2DD which will issue that message and terminate with RC=8.

Otherwise, on return from the exit, CMNDB2DD will re-read the SQL from the DDLIN ddname (i.e. including any changes made by the exit) and will continue to process the modified SQL sentence as directed by the other sysin parameters.

The 'data interface' to this exit is as follows. Shown here in COBOL and PL/I copybook format, and equivalent REXX variable names:

## COBOL copybook

```
01 DB2D.
***
* HLL DATA ITEMS FOR CMNDB2DD CALLED EXIT
***
03 DB2DGO          PIC X(3).
*                  PROCEED? YES/NO
03 DB2DMSG         PIC X(128).
*                  MESSAGE
03 DB2DACTN        PIC X(8).
*                  CURRENT ACTION
03 DB2DDBZS        PIC X(4).
*                  DB2 SUBSYSTEM ID
03 DB2DZMFI        PIC X(4).
*                  ZMF SUBSYSTEM ID
03 DB2DZPKG        PIC X(10).
*                  ZMF PACKAGE
03 DB2DEXTN        PIC X(8).
*                  EXTERNAL EXIT NAME
03 DB2DCMPN        PIC X(8).
*                  COMPONENT
03 DB2DUSER        PIC X(8).
*                  USERID
03 DB2DSLLOC       PIC X(128).
*                  SRC LOCN TEMPLATE
03 DB2DTLOC        PIC X(128).
*                  TGT LOCN TEMPLATE
03 DB2DDLLOC       PIC X(128).
*                  DEPLOY FROM LOCATION
03 DB2DSCOL        PIC X(128).
*                  SRC COLLID TEMPLATE
03 DB2DTCOL        PIC X(128).
*                  TGT COLLID TEMPLATE
03 DB2DSSCM        PIC X(128).
*                  SRC SCHEMA TEMPLATE
03 DB2DTSCM        PIC X(128).
*                  TGT SCHEMA TEMPLATE
03 DB2DSAUT        PIC X(128).
*                  SRC AUTHID TEMPLATE
03 DB2DTAUT        PIC X(128).
*                  TGT AUTHID TEMPLATE
03 DB2DSQUL        PIC X(128).
*                  SRC QUAL TEMPLATE
03 DB2DTQUL        PIC X(128).
*                  TGT QUAL TEMPLATE
03 DB2DDQUL        PIC X(128).
*                  DEPLOY QUAL
03 DB2DSOWN        PIC X(128).
*                  SRC OWNER TEMPLATE
03 DB2DTOWN        PIC X(128).
*                  TGT OWNER TEMPLATE
03 DB2DDOWN        PIC X(128).
*                  DEPLOY OWNER
03 DB2DSWLM        PIC X(54).
*                  SRC WLM TEMPLATE
03 DB2DTWLM        PIC X(54).
*                  TGT WLM TEMPLATE
```

## PL/I copybook

```
/*THE API DATA AREA MAP FOLLOWS */
DCL 1 DB2D,
  2 DB2DGO      CHAR(3),      /*PROCEED? YES/NO */
  2 DB2DMSG     CHAR(128),    /*MESSAGE          */
  2 DB2DACTN    CHAR(8),      /*CURRENT ACTION   */
  2 DB2DDB2S    CHAR(4),      /*DB2 SUBSYSTEM ID */
  2 DB2DZMFI    CHAR(4),      /*ZMF SUBSYSTEM ID */
  2 DB2DZPKG    CHAR(10),     /*ZMF PACKAGE      */
  2 DB2DXTN     CHAR(8),      /*EXTERNAL EXIT NAME */
  2 DB2DCMPN    CHAR(8),      /*COMPONENT        */
  2 DB2DUSER    CHAR(8),      /*USERID           */
  2 DB2DSLLOC   CHAR(128),    /*SRC LOCN TEMPLATE */
  2 DB2DTLOC    CHAR(128),    /*TGT LOCN TEMPLATE */
  2 DB2DDLLOC   CHAR(128),    /*DEPLOY FROM LOCATION */
  2 DB2DSCOL    CHAR(128),    /*SRC COLLID TEMPLATE */
  2 DB2DTCOL    CHAR(128),    /*TGT COLLID TEMPLATE */
  2 DB2DSSCM    CHAR(128),    /*SRC SCHEMA TEMPLATE */
  2 DB2DTSKM    CHAR(128),    /*TGT SCHEMA TEMPLATE */
  2 DB2DSAUT    CHAR(128),    /*SRC AUTHID TEMPLATE */
  2 DB2DTAUT    CHAR(128),    /*TGT AUTHID TEMPLATE */
  2 DB2DSQUL    CHAR(128),    /*SRC QUAL TEMPLATE */
  2 DB2DTQUL    CHAR(128),    /*TGT QUAL TEMPLATE */
  2 DB2DDQUL    CHAR(128),    /*DEPLOY QUAL      */
  2 DB2DSOWN    CHAR(128),    /*SRC OWNER TEMPLATE */
  2 DB2DTOWN    CHAR(128),    /*TGT OWNER TEMPLATE */
  2 DB2DDOWN    CHAR(128),    /*DEPLOY OWNER     */
  2 DB2DSWLM    CHAR(54),     /*SRC WLM TEMPLATE */
  2 DB2DTWLM    CHAR(54);     /*TGT WLM TEMPLATE */
```

## REXX variable names

```
proceed
messageText
currentAction
db2Subsystem
zmfSubsystemId
zmfPackage
externalName
component
userId
srcLocationTemplate
tgtLocationTemplate
deployFromLocation
srcCollIdTemplate
tgtCollIdTemplate
srcSchemaTemplate
tgtSchemaTemplate
srcAuthIdTemplate
tgtAuthIdTemplate
srcQualifierTemplate
tgtQualifierTemplate
deployQualifier
srcOwnerTemplate
tgtOwnerTemplate
deployOwner
srcWlmTemplate
tgtWlmTemplate
```

## REXX Rample

```
Sample REXX showing the changing of the version to be the package id:
/* REXX          */
/* */
/* Show all CMNDB2DD API fields */
/* Show DDL/SQL on input */
/* Change the version to be the ZMF package id */
/* Show DDL/SQL passed back to CMNDB2DD */
/* */

proceed = "YES"

say " "
say "-----"
say " CMNDB2DD HLLX "
say "-----"
say " "
say "currentAction      : "currentAction
say "db2Subsystem       : "db2Subsystem
say "zmfSubsystemId     : "zmfSubsystemId
say "zmfPackage         : "zmfPackage
say "externalName       : "externalName
say "component          : "component
say "srcLocationTemplate : "srcLocationTemplate
say "tgtLocationTemplate : "tgtLocationTemplate
say "deployFromLocation : "deployFromLocation
say "srcCollIdTemplate   : "srcCollIdTemplate
say "tgtCollIdTemplate   : "tgtCollIdTemplate
say "srcSchemaTemplate   : "srcSchemaTemplate
say "tgtSchemaTemplate   : "tgtSchemaTemplate
say "srcAuthIdTemplate   : "srcAuthIdTemplate
say "tgtAuthIdTemplate   : "tgtAuthIdTemplate
say "srcQualifierTemplate : "srcQualifierTemplate
say "tgtQualifierTemplate : "tgtQualifierTemplate
say "deployQualifier     : "deployQualifier
say "srcOwnerTemplate    : "srcOwnerTemplate
say "tgtOwnerTemplate    : "tgtOwnerTemplate
say "deployOwner         : "deployOwner
say "srcWlmTemplate      : "srcWlmTemplate
say "tgtWlmTemplate      : "tgtWlmTemplate
say " "
"execio * diskr DDLIN (stem ddl. finis"

say " "
say "+-----+"
say "| DDL/SQL as supplied to HLL exit follows |"
say "+-----+"
say " "
Do i = 1 to ddl.0
    Say ">"ddl.i"<"
End

Do i = 1 to ddl.0

    ix = pos('VERSION ',ddl.i)
    if ix = 0 then iterate

    ix = ix + 8
    endofit = substr(ddl.i,ix)
    versn   = subword(endofit,1,1)
    pastver = subword(endofit,2)
    replace = substr(ddl.i,1,ix-1) || zmfPackage || " " || pastver || '25'x
    ddl.i = replace

End
```

```
messageText = "VERSION has been set to package id."
"execio * diskw DDLIN (stem ddl. finis"

say " "
say "+-----+"
say "| DDL/SQL as modified by HLL exit follows |"
say "+-----+"
say " "
Do i = 1 to ddl.0
  Say ">"ddl.i"<"
End

exit 0
```



# 11. ISPF Tables and Variables

---

The following pages describe ISPF Dialog Manager tables and variables used in the ChangeMan ZMF Db2 Option.

- [ISPF Tables](#)
- [Single Entry Control Variables](#)

## ISPF Tables and Variables

---

The following pages describe ISPF Dialog Manager tables and variables used in the ChangeMan ZMF Db2 Option.

### ISPF Tables

This section lists the ISPF tables used in file tailoring to create Db2 processing steps in jobs for promote, demote, install, backout, baseline ripple, and reverse baseline ripple.

For the list of variables in each of these tables, search for the table name in member #VARLIST in the CMNZMF SKELS library.

#### **Caution**

ZMF skeletons do not use these ISPF tables consistently across different package processes such as promotion and installation. Before you apply Db2 Option customization designed for one package process to skeletons for a different package process, examine the delivered skeletons to see that you are using the appropriate ISPF tables.

### CMNDB2S1 - Db2 Physical Subsystem Table

Each row in table CMNDB2S1 describes a physical subsystem defined for the ZMF instance. All physical subsystems for the ZMF instance are included in this table.

Values for the ISPF variables in this table are set on global administration panels for the Db2 option. These panels include:

- **Db2 Physical Subsystems - Part 1 of 2** (CMNGD2S0)
- **Db2 Physical Subsystems - Part 2 of 2** (CMNGD2S1)

## CMNDB2SS - Db2 Physical Subsystem Table for BIND

Table CMNDB2SS contains only those physical subsystems where Db2 BIND processing is required by the current action. Each row is initially formed from equivalent rows in CMNDB2S1. That is, when a component will be copied into a library that is listed in the Db2 BIND active library table CMNDB2AL, the logical subsystem listed for the active library is used to get a logical subsystem definition, which contains a physical subsystem that is used to copy a row from CMNDB2S1 to CMNDB2SS.

However, information about bind failure significance is also gathered from these logical subsystems. The "bind failure significance" variable (STBINDF) forms a key for this table so that you may have up to two rows for each physical subsystem. (STBINDF can be "YES" or "NO.")

This structure allows CMNDB2SS to be used in the skeletons to generate zero, one, or two CMNDB2PL/BIND step couplets for each physical subsystem, depending on whether:

- No logical subsystems active in this promote/demote target this physical subsystem (zero).
- All such logical subsystems have the same setting for bind failure significance (one).
- There is a mix of bind failure significance settings (two).

## CMNDB2S2 - Db2 Physical Subsystem Table (With Bind Failure)

This table has the same global scope as CMNDB2S1 but it is constructed with "bind failure" information (STBINDF), in a similar way to CMNDB2SS.

There are currently no delivered skeletons which make use of this table but it is available for use. CMNDB2S2 has been chosen so as not to upset the existing use of CMNDB2S1.

## CMNDB2N1 - Db2 Logical Subsystem Table

Each row in table CMNDB2N1 describes a logical subsystem defined in the application that is being processed in file tailoring. All logical subsystems for the application are included in this table.

Values for the ISPF variables in this table are set on application administration panels for the Db2 option. These panels include:

- **Db2 Logical Subsystems** (CMNLD2LN)
- **Db2 Logical Subsystem logical subsys Parameter Settings** (CMNGD2PM)
- **Db2 Logical Subsystem logical subsys Templates** (CMNGD2L2)

## CMNDB2NN - Db2 Logical Subsystem Table for BIND

---

Table CMNDB2NN contains a subset of the rows in table CMNDB2N1. The information for a logical subsystem is identical in both tables, but CMNDB2NN contains only those logical subsystems where Db2 BIND processing is required.

When a component will be copied into a library that is listed in the Db2 BIND active library table CMNDB2AL, the logical subsystem listed for the active library is used to copy a row from table CMNDB2N1 into table CMNDB2NN.

## CMNDB22N - Db2 Logical Subsystem Table for Secondary BIND

---

Table CMNDB22N contains rows describing secondary bind requirements for this action. These requirements are defined using application admin in the ZMF Db2 option.

The primary bind logical subsystem is assigned when a component will be copied into a library that is listed in the Db2 BIND active library table CMNDB2AL.

If there are one or more secondary bind definitions linking the primary bind logical subsystem to other logical subsystems then a row will be written to this table for each such association.

The table is similar to CMNDB2NN in content but has added fields for the secondary bind process (see list below).

In the sample skeletons this table is used during file tailoring to generate CMNDB2PL/ BIND steps to address secondary bind requirements in the same job as the primary bind.

TABLE NAME: CMNDB22N DESCRIPTION: Secondary binds for plan/pkg

NOTE: The field values are those that apply to the secondary logical subsystem. This table can be used to drive binds for the secondary logical subsystem if binds have already been generated for the primary logical subsystem.

TABLE COLUMN	VARIABLE NAME	VARIABLE LENGTH	VARIABLE DESCRIPTION
01	NT2NAM	08	Primary logical subsystem name
02	NT2COLT	128	Primary Pkg ID target template
03	NT2COLS	128	Primary Pkg ID source template
04	NTNNAM	08	Secondary logical subsys name
05	NTSSYS	04	Db2 logical subsystem ID
06	NTPVLV	02	Associated promotion level
07	NTPNME	08	Associated promotion name
08	NTBDID	128	Db2 insert bind owner ID
09	NTTOTPT	128	Db2 owner target template
10	NTSOTPT	128	Db2 owner source template
11	NTTMPT	24	Db2 PLAN target template
12	NTSTMPT	24	Db2 PLAN source template
13	NTPTMP	128	Db2 PKG ID target template
14	NTSPTMP	128	Db2 PKG ID source template
15	NTTLTPT	128	Db2 location ID target template
16	NTSLTPT	128	Db2 location ID source template
17	NTQUAL	128	Db2 qualifier
18	NTSQTPT	128	Db2 qualifier source template
19	NTQTPT	128	Db2 qualifier target template
20	NTBINF	03	Fail action if bind fails
21	NTREMT	08	Db2 logical remote site
22	NTPRMLV	02	Associated promotion level
23	NTPRMNM	08	Associated promotion name
24	NTDBR	03	Preferred DBRM libtype

## CMNDB2NQ - Db2 Logical Subsystem Table for SQL/Stored Procedure

Table CMNDB2NQ contains a subset of the rows in table CMNDB2N1. The information for a logical subsystem is identical in both tables, but CMNDB2NQ contains only those logical subsystems where SQL/stored procedure processing is required.

When a component will be copied into a library that is listed in the SQL/stored procedure active library table CMNSQLAL, the logical subsystem listed for the active library is used to copy a row from table CMNDB2N1 into table CMNDB2NQ.

## CMNDB2AL - Db2 BIND Active Library Table

This table lists the Db2 Option active libraries for BIND processing in the application being processed in file tailoring.

Values for the ISPF variables in this table are set in application administration for the Db2 Option on the **Db2 Active Library List** panel (CMNLD2AL). Table CMNDB2AL contains only those active libraries where the **BIND/SQL** field is set to **B**.

## CMNSQLAL - Db2 SQL/Stored Procedure Active Library Table

This table lists the Db2 Option active libraries for SQL/stored procedure processing in the application being processed in file tailoring.

Values for the ISPF variables in this table are set in application administration for the Db2 Option on the **Db2 Active Library List** panel (CMNLD2AL). Table CMNSQLAL contains only those active libraries where the **BIND/SQL** field is set to **S**.

## CMNSQLTK - Db2 General Token Table for SQL/DDDL operations

This table lists the token name (64), token source template (128) and token target template (128).

## Single Entry Control Variables

This table describes key ISPF variables that control file tailoring for the Db2 Option. For other variables used in file tailoring for the Db2 Option, see skeleton CMN\$\$VAR and member #VARLIST in the CMNZMF SKELS library.

Field	Definition
RUNDB2PL	Set to YES when an active library has been targeted by load or BIND components, and there are staged BIND or DBRM components in the change package.
DBBSTG	Set to YES when a change package contains a BIND component.
DBRSTG	Set to YES by when a change package contains a DBRM component.
PKGSTG	Set to YES when a change package contains a BIND PACKAGE component.
PKGTYPE	Set to the library type in Db2 Option administration with Db2 Subtype P.

<b>Field</b>	<b>Definition</b>
REBPKG	Set to Y if there are load members in the change package related to source in component history that have status BAS and the Db2 Precompile indicator set to Y.
BIND2ND	Set to Y if secondary bind requirements exist for this primary bind.

# 12. Transaction Codes

---

- Detailed Job List
- Miscellaneous Transactions - at Either Site

## Detailed Job List

---

Each job is part of specific transactions that are all created for a change package during Freeze processing. The transaction name (member name of Job JCL Library) is built using the application mnemonic, a transaction code, and the last 2 or 3 digits of the change package number. Thus, job 10 for change package number ABC 20, would be named:

ABC10020. The following table shows the detailed job list.

DEVELOPMENT CENTER		REMOTE SITE	
Job	Action	Job	Action
	Package is audited and/or frozen. Jobs are created in "...X.&node". Package is approved. Job 10 is submitted to initiate the distribution.		
10	CMNBATCH transaction '10' says distribution initiated and status is changed to DIS. Vehicle is asked to submit job 11 at remote site.		
11	Staging libraries are sent to remote site.	10	Staging libraries are received including QSAM package master. Job 11 is submitted.

DEVELOPMENT CENTER	REMOTE SITE		
		11	CMNBATCH transaction '11' overlays package records (on PM) with QSAM package master; proper node record is time stamped; status is DIS. Job 14 is submitted (only if IEBCOPY is not used).
		14	Job 14 requests vehicle to submit 15 at DEV site.
		17	Provided the scheduler is external, submits job 20 (permanent package) or job 30T (temporary package).
		18	Requests vehicle to submit 19 at DEV site.
15	Job 15 is submitted (only if IEBCOPY is used).		
15	CMNBATCH transaction '15' stamps acknowledgment of distribution.		
19	Notification to the user specified in the <code>Notify user</code> field that distribution failed.		
		21	Perform Db2 bind for production installation (INSTALL IN PROD = YES).
		20	Submitted to check if package was previously installed, if not, then it begins installation.



DEVELOPMENT CENTER	REMOTE SITE		
		20	CMNBATCH transaction '20' changes package status to 'INS'.
		20	Job 24 is submitted. (Only if IEBCOPY is not used.)
		20t	If Temporary, Job 20t runs to install members into Temporary libraries.
		24	Requests vehicle to submit 25 at DEV site.
		28	Requests vehicle to submit 29 at DEV site.
25	CMNBATCH transaction '25' changes package status to 'INS'.		
29	Notification to the user specified in the <code>Notify user</code> field that the installation failed.		
25	If Permanent, Job 30 is submitted.		
		30	Submitted if system environment is 'ALL'.
30	CMNBATCH transaction '30' changes package status to 'BAS' and ripples the baseline.		
30	Delete members from Promotion Libraries based on promotion level and library type.		

DEVELOPMENT CENTER	REMOTE SITE		
		31	If Temporary, Job 31 runs to delete members from Temporary libraries.
		31t	CMNBATCH transaction '31' changes package status to 'TCC' (Temporary Change Cycled) and date/time stamp. Submit job 35.
		32	Performs Db2 bind for production installation (INSTALL IN PROD = NO).
		34t	Requests vehicle to submit 35t at DEV site.
35t	Package status updated to TCC and date/time stamped when all remote sites have been cycled.		
		38t	Requests vehicle to submit 39t at DEV site.
39t	Notification to the user specified in the <code>Notify user</code> field that the package cycle failed.		
			CASE: A permanent change must be backed out. Operator makes human decision to back out (full) particular package. Enters backout reasons on panel and Started Task copies package to same flat file that was sent from development center. Job 50 is submitted.

DEVELOPMENT CENTER	REMOTE SITE		
		49	Job 49 runs the Db2 bind for production backout (INSTALL IN PROD = YES).
		50	Backs out the change by copying back from BKUP Libraries. Changes package status to 'BAK'. Job 54 is submitted if IEBCOPY is used, else job 51.
		50	If system environment is 'ALL', job 55 is submitted.
		51	Job 51 transmits a QSAM package master to the development center and requests a vehicle to submit job 54.
54	Reads flat package and transmits reasons; updates backout reasons into correct package.		
55	Job 55 is submitted to reverse ripple the Baseline if all remote sites are backed out.		
55	Status is changed to 'BAK', * node record is date and time stamped.		
		56	Job 56 runs the Db2 bind for production backout (INSTALL IN PROD = NO).
		58	Job 58 requests vehicle to submit 59 at DEV site.

DEVELOPMENT CENTER	REMOTE SITE
59	Notification to the user specified in the <code>Notify user</code> field that the package backout failed.
	64 Job 64 requests vehicle to submit 65 at DEV site.
	Package is audited and/or frozen. Jobs are created in "...X.&node". Package is approved. Job 10 is submitted to initiate the distribution.

## Miscellaneous Transactions - at Either Site

The following table shows the miscellaneous transactions.

Transaction	Explanation
CMNBATCH transaction '05'	Submits a job based on: STE=site NOD=node SUB=jobname
CMNBATCH transaction '65'	Reverts package back to development: Reset general component freeze flag. Reset all major date/time stamps. Set revert date/time stamp at remote site.
CMNBATCH transaction '80'	Promotes or demotes a package. Checks out components with or without package association.
CMNBATCH transaction '90'	Activates a component.
CMNBATCH transaction '92'	Deletes Staging libraries.

<b>Transaction</b>	<b>Explanation</b>
CMNBATCH transaction '93'	Synchronizes the implementation calendar.
CMNBATCH transaction '94'	Deletes Change Package records.
CMNBATCH transaction '96'	Decrements the Implementation Calendar when packages are deleted.
CMNBATCH transaction '99'	This transaction is invoked to notify the user any time there is a job failure.

# 13. Examples

---

- Native SQL SP Versions and Bind Deploy
- Support Use of zFS File Type for SP Components

## Native SQL SP Versions and Bind Deploy

---

The following are some screen shots designed to show various aspects of the Native SQL stored procedure support using Db2 versioning and the BIND DEPLOY distribution methodology.

Note that the 'standard' processes associated with the drop/create methodology are similar to the existing DDL processes. There are added facilities such as the HLL exit point and the PASSTHRU facility which are described elsewhere in this document.

This section is intended to help you get to grips with support for Db2 versioning and BIND DEPLOY.

In this example, we are using Data Studio to generate/change the stored procedure definitions/SQL. Data Studio then deploys to our target Db2 subsystem (in this case this is DSN on U001).

We then stage from the Db2 catalog directly into a ZMF package and proceed to promote to a local site (using Db2 versioning but not bind deploy) and to remote site (using both Db2 versioning and bind deploy). The install to production uses versioning/bind deploy and we install to U810DP (which is the DSN Db2 subsystem) and U810P (which is for the DSN1 subsystem).

The library type for our Native SQL SP components is NSQ.

Here are some admin definitions.

First, the local site promote library. Note that the 'Cleanup Level' flag is set to 'N'. It is assumed that users will not want their promoted SPs to be dropped automatically as there is no concept of a search hierarchy for SPs in the same way as their might be for load modules.

When we promote to this level we will copy the SP component (which is a PDS member) to the library shown.

```

CMNLRPM3          STEV/LOCALVER - Promotion Libraries          Row 1 to 1 of 1
Command ==> _____ Scroll ==> CSR

Promotion name:  UNIT      Level: 10

      Syslib Cleanup
      Lib exclude Level Target libraries
_____ NSQ   Y      N   ZMFSD.VUNIT.NSQ          + Shadow
                                   ZMFSD.VUNIT.NSQ          + Library 1
                                   _____          + Library 2
                                   _____          + Library 3

```

Enter N in this field if cleanup is to be skipped for this library type at this level during a promotion or the installation of a package.

Similarly, for the remote site promotion:

```

CMNLRPM3          STEV/REMOTEVER - Promotion Libraries          Row 1 to 1 of 1
Command ==> _____ Scroll ==> CSR

Promotion name:  UNIT      Level: 10

      Syslib Cleanup
      Lib exclude Level Target libraries
_____ NSQ   Y      N   ZMFSD.VQA1.NSQ          + Shadow
                                   ZMFSD.VQA1.NSQ          + Library 1
                                   _____          + Library 2
                                   _____          + Library 3

```

And here are the production library definitions for sites U810DP (the local DP site) and U810P (the 'remote' P site - not really remote) - U810DP:

```

CMNCPRDL          STEV - U810DP Production Libraries          Row 5 to 10 of 10
Command ==> _____ Scroll ==> CSR

      Type Production dataset name +
      Temporary dataset name +
      Backup dataset name +
_____ NSQ ZMFSD.PROD.NSQ
      NULLFILE
      ZMFSD.PROD.NSQ.BKUP
...

```

and U810P:

```

CMNCPRDL          STEV - U810P Production Libraries          Row 2 to 3 of 3
Command ==> _____ Scroll ==> CSR

Type Production dataset name +
      Temporary dataset name +
      Backup dataset name    +
_____ NSQ ZMFSD.PROD1.NSQ
          NULLFILE
          ZMFSD.PROD1.NSQ.BKUP
...

```

None of the above is new, just standard ZMF admin.

Now, turning to the Db2 option admin. Here is the list of logical subsystems we are working with. We have one logical subsystem per target 'environment'

```

CMNLD2LN          Db2 Logical Subsystems          Row 1 to 4 of 4
Command ==> _____ Scroll ==> CSR

Line commands:
  P Specify miscellaneous parameters
  T Specify BIND process variable templates
  Q Specify SQL process named variable templates
  G Specify SQL process general token variable templates

      Logical   Db2
      name      subsys   Site      Description
_____ PRODV    DSN      U810DP   PROD USING VERSION + DEPLOY
_____ PRODV1   DSN1     U810P    PROD ON DSN1 USING VER/DEPLOY
_____ QAD1     DSN1     REMOTVER  QA ON DSN1 USING VER/DEPLOY
_____ UNITV   DSN      LOCALVER  UNIT TEST WITH VERSION

```

All of these logical subsystems have the following 'Use Db2 versioning' set in the 'miscellaneous parameters'. Without this you will be using the standard Drop/Create paradigm.

```

CMNGD2PM          Db2 Logical Subsystem UNITV Settings
Command ==> _____

Preferred Libtypes:
DBRM . . . . . DBR
Plan bind parameters . . . . . DBB
Package bind parameters . . . . . PKG

General Parameters:
Enter "/" to select option
_ Bind Failure is significant
_ Recycle Stored Procedures where WLM Environment is . .
_ Maintain Trigger Sequence
/ Use Db2 versioning for Native-SQL Stored Procedures

```

Here are what the different logical subsystems have specified for the SQL templates.



Note that most of these values are used to generate the names used at the target subsystem. So, in the case of UNITV below, the stored procedures will have whatever schema they have specified in the code replaced with 'UNIT', i.e. UNIT.spname

The DEPLOY location (highlighted in red) is slightly different in that it used when the logical subsystem is defined as the 'source' subsystem for the deploy action. More about this below but, in effect, the bind deploy request is sent to the source subsystem location as specified here.

In our test subsystems we had the following locations defined:

DSN	DB2V11
DSN1	SOW1DSN1

```

CMNGD2L3          Db2 Logical Subsystem UNITV SQL Process Templates (Named)
Command ==> _____

  Templates          Target          Source          Deploy

Schema . . . . . UNIT~ ~ ~ ~ ~ + _____ +
Collection . . . . . UNIT~ ~ ~ ~ ~ + _____ +
WLM . . . . . _____ + _____ +

Location . . . . . _____ + _____ + DB2V11 +
Qualifier . . . . . UNIT~ ~ ~ ~ ~ + _____ + UNIT +
Owner . . . . . SERD~ ~ ~ ~ ~ + _____ + SERD +
  
```

```

CMNGD2L3          Db2 Logical Subsystem QAD1 SQL Process Templates (Named)
Command ==> _____

  Templates          Target          Source          Deploy

Schema . . . . . QA~ ~ ~ ~ ~ + _____ +
Collection . . . . . QA~ ~ ~ ~ ~ + _____ +
WLM . . . . . _____ + _____ +

Location . . . . . _____ + _____ + DB2V11 +
Qualifier . . . . . QA~ ~ ~ ~ ~ + _____ + QA +
Owner . . . . . SERD~ ~ ~ ~ ~ + _____ + SERD +
  
```

```

CMNGD2L3          Db2 Logical Subsystem PRODD SQL Process Templates (Named)
Command ==> _____

  Templates          Target          Source          Deploy

Schema . . . . . PROD~ ~ ~ ~ ~ + _____ +
Collection . . . . . PROD~ ~ ~ ~ ~ + _____ +
WLM . . . . . _____ + _____ +

Location . . . . . _____ + _____ + DB2V11 +
Qualifier . . . . . PROD~ ~ ~ ~ ~ + _____ + PROD +
Owner . . . . . SERD~ ~ ~ ~ ~ + _____ + SERD +

```

```

CMNGD2L3          Db2 Logical Subsystem PRODD1 SQL Process Templates (Named)
Command ==> _____

  Templates          Target          Source          Deploy

Schema . . . . . PROD~ ~ ~ ~ ~ + _____ +
Collection . . . . . PROD~ ~ ~ ~ ~ + _____ +
WLM . . . . . _____ + _____ +

Location . . . . . _____ + _____ + DB2V11 +
Qualifier . . . . . PROD~ ~ ~ ~ ~ + _____ + PROD +
Owner . . . . . SERD~ ~ ~ ~ ~ + _____ + SERD +

```

Having defined our logical subsystems we now need to tell the various ZMF actions when to use them. This is done (as usual) using the active library definitions. The first one below says that when ZMF delivers to ZMFSD.VUNIT.NSQ then Db2 actions will be governed by the UNITV logical subsystem. In this case we will create/alter a version of the SP and then activate that new version, according to the templates/values associated with the UNITV logical subsystem.

The others all target something called a connector, rather than a traditional logical subsystem. If an active library targets a connector you are requesting the distribution to be performed via BIND DEPLOY (see below for more on connectors).

```

CMNLD2AL          Db2 Active Library List          Row 1 to 4 of 4
Command ==> _____          Scroll ==> CSR

  Logical   Bind   Db2 active library name
  name     /SQL
  _____
  UNITV    S      ZMFSD.VUNIT.NSQ
  UNIT2QA1 S      ZMFSD.VQA1.NSQ
  UNIT2PRD S      ZMFSD.PROD.NSQ
  UNIT2PR1 S      ZMFSD.PROD1.NSQ

```

There is a new Db2 subtype which is used to indicate that we are processing a Native SQL stored procedure (NSQ):

```

CMNDLLT0          STEV - Db2 Library Types          Row 1 to 11 of 11
Command ==> _____ Scroll ==> CSR

Lib              Sub      End SQL
type            Description  type      sentence
DBR              DB2 DBRM's          R          -
DBP              DB2 Bind Package Commands P          -
PKG              DB2 Package Bind Control P          -
DBB              DB2 Bind Plan Commands P          -
STL              DB2 External Stored Procedure Load S          -
XPQ              DB2 External SQL stored proc Source D          #
SPQ              Native SQL Stored Procedure -          -
MPQ              Native SQL Stored Procedure metadata -          -
SPD              DB2 Stored Procedure Definition D          #
NSQ              Native SQL Stored Procedures N          #
DDL              Data Definition Language D          @
  
```

To use BIND DEPLOY we have to define connectors. These connect two different logical subsystems as source and target for a BIND DEPLOY operation. The source values are used to find the pre-existing stored procedure and the target values tell Db2 how to define the copy of the stored procedure at the target location. In this test we have 2 connectors, all deploying from the UNITV logical subsystem to 3 different targets.

```

CMNDL2CL          Logical Subsystem Connectors for Appl - STEV          Row 1 to 3 of 3
Command ==> _____ Scroll ==> CSR

Connector        Source      Target
name             name       name       Description
-----
UNIT2PRD         UNITV      PRODD      UNIT TO PROD (DSN)
UNIT2PR1         UNITV      PRODD1     UNIT TO PROD (DSN1)
UNIT2QA1         UNITV      QAD1       UNIT TO QA (DSN1)
***** Bottom of data *****
  
```

When you define a connector all you are doing is specifying the source and target logical subsystem names, along with a description for it. There is nothing more you can define. Once you have saved the connector away, if you select the definition from the list you will get a (output only) indication of the values which will be used by the BIND DEPLOY operation. For example, for UNIT2PRD we see these values:

```

CMNLD2CN          Application STEV Logical Subsystem Connector - UNIT2PRD
Command ==> _____

                UNIT TO PROD (DSN)

Source . . . . UNITV
Subsystem id . DSN
Location . . . DB2V11          +

Templates      Target          Source
Schema . . . . UNIT~~~~~ +          +

Target . . . . PRODD
Subsystem id . DSN
Site . . . . U810DP

Templates      Target          Source          Deploy
Collection . . PROD~~~~~ +          +
Qualifier . . PROD~~~~~ +          + PROD          +
Owner . . . . SERD~~~~~ +          + SERD          +

```

The source location for the bind deploy is DB2V11 (which is hosted by subsystem DSN) and the schema used to define the SP at that location is, in this case, UNIT.

The target location for the bind deploy will be determined by ZMF when the process runs but the logical subsystem we are using is PRODD (which is defined for site U810DP and hosted by the DSN Db2 subsystem). We will use PROD as the schema for the target SP and use a qualifier of PROD and an owner of SERD.

All of these values are taken from either the source or target logical subsystem definition as specified for this connector.

Here are the values which will be used by the other connectors in this example:

```

CMNLD2CN          Application STEV Logical Subsystem Connector - UNIT2PR1
Command ==> _____

                UNIT TO PROD (DSN1)

Source . . . . UNITV
Subsystem id . DSN
Location . . . DB2V11          +

Templates      Target          Source
Schema . . . . UNIT~~~~~ +          +

Target . . . . PRODD1
Subsystem id . DSN1
Site . . . . U810P

```

Templates	Target	Source	Deploy
Collection . . .	PROD	+	+
Qualifier . . .	PROD	+	+ PROD
Owner . . . .	SERD	+	+ SERD

```

CMNLD2CN          Application STEV Logical Subsystem Connector - UNIT2QA1
Command ===> _____

                UNIT TO QA (DSN1)

Source . . . . UNITV
Subsystem id . DSN
Location . . . DB2V11          +

Templates        Target          Source
Schema . . . . UNIT              +

Target . . . . PRODD1
Subsystem id . DSN1
Site . . . . REMOTVER

Templates        Target          Source          Deploy
Collection . . QA              +
Qualifier . . QA              + QA
Owner . . . . SERD              + SERD

```

That's it for admin. Most of it is no different to the kind of Db2 option admin which preexisted. The major differences being the 'Use Db2 versioning' setting in miscellaneous parameters and the whole area of connectors being used to define BIND DEPLOY usage.

Now to see some output from the lifecycle processes. We have used Data Studio to generate a change to a pre-existing stored procedure. Data Studio runs its own proprietary deployment method (consisting of OCO stored procedures) so we let it complete deployment and pick up the SP from the target Db2 catalog.

In this example we are generating a stored procedure which has a name longer than 8 bytes (LONG\_NAME\_SP\_NUMBER\_3), the max length for the name is 128 bytes. Note that, the ZMF package component for this SP will be a member name which, of course, is restricted to 8 bytes. To make things consistent it is best to keep the SP name the same as the component name. However, the only downside to this is that there is no formal tieup between the component name and the SP name.

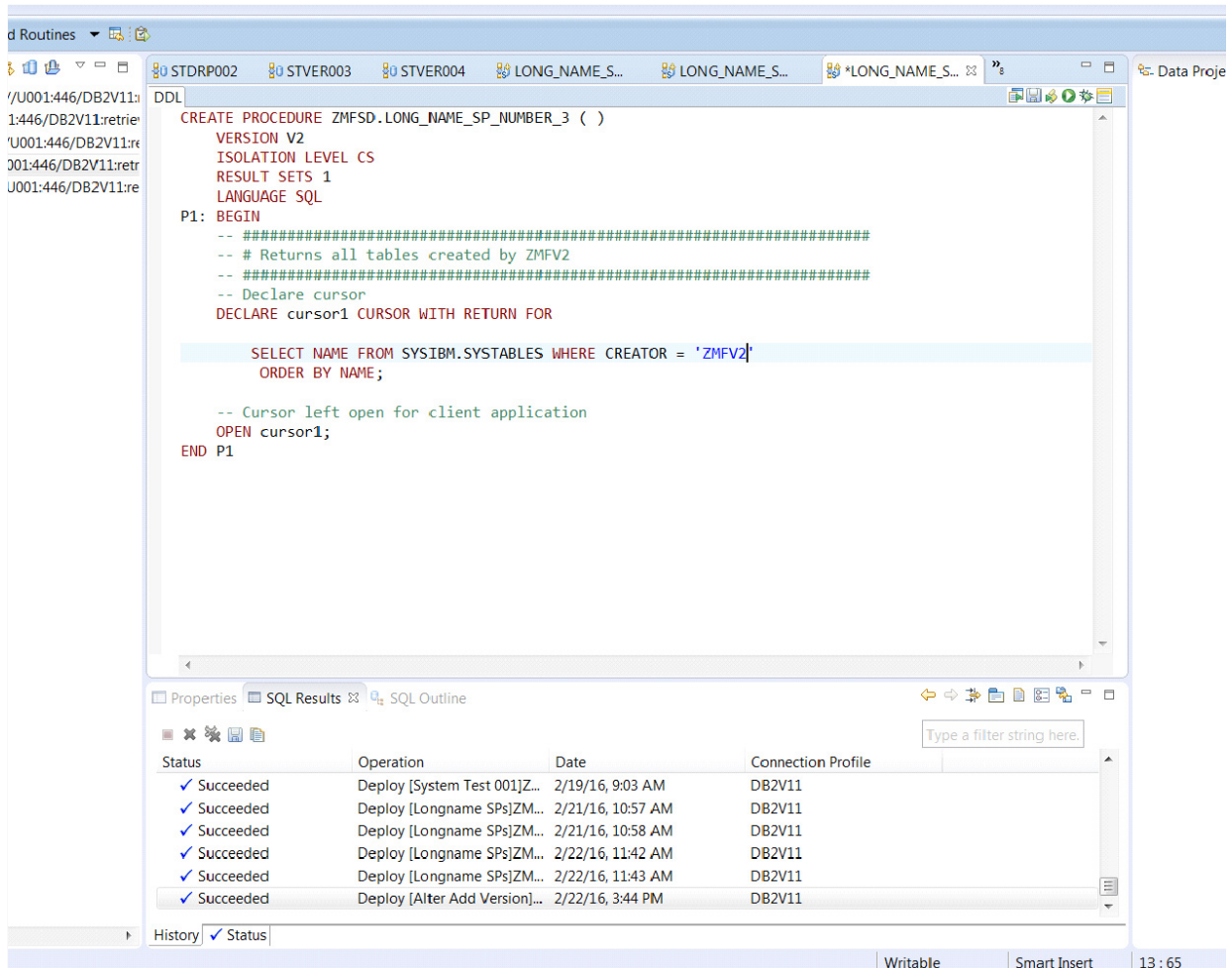
The SP presented here is a simple example (we are interested in the process not the SP itself). Here we are about to deploy version V2 to our target development Db2 subsystem.

Two things of note here:

1. The version identifier is a freeform 64 byte field (122 bytes if you use DBCS). There is no ordinal sense to this field. While the default first version of an SP is assigned the version V1 by

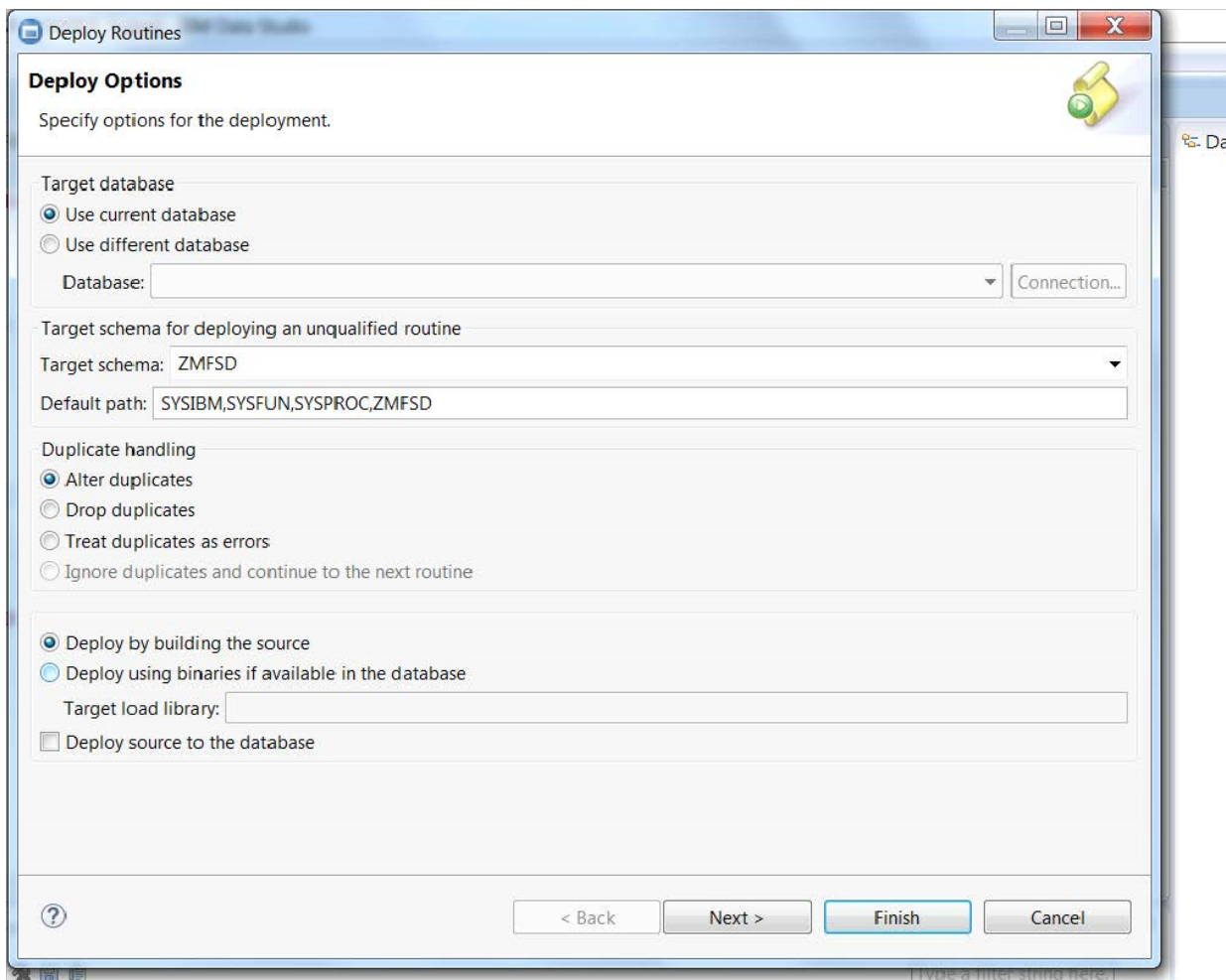
Db2 and we are adding here a new version of the same stored procedure called V2, there is no ordinal meaning to V1 and V2.

2. The SP has been given a schema (ZMFSD, my userid, in this case). This is important to allow the lifecycle templating to work



The deployment mechanism gives you various options. The one of interest here is the 'Alter duplicates' flag. As far as Db2 is concerned there is a single entity (i.e. stored procedure) called ZMFSD.LONG\_NAME\_SP\_NUMBER\_3 at the target Db2 subsystem. This deployment is attempting to create another version of the same item, this is a duplicate item. Setting the 'Alter duplicates' flag tells the deployment to change the DDL (which, at the moment, say CREATE PROCEDURE) to ALTER PROCEDURE ... ADD VERSION ...

This is what we want for this example which makes use of Db2 versioning of Native SQL SPs



At this point the SP is in the target Db2 catalog. Because this is a change to an existing ZMF managed component we need to check it out into our target package before we can stage the changed version. This is where having the same name for the component as the SP comes in useful. Because we have an SP name longer than 8 bytes then we have to know the name of the component used to 'shadow' the SP in the Db2 catalog. In this case it is LONGSP03.NSQ.

Once LONGSP03.NSQ has been checked out we can then stage the new version which Data Studio has deployed to a local Db2 subsystem (DSN in our case).

Using S1 against the package gives us the familiar stage-from-development panel, with a new option.

```

CMNSTG02                               Stage from Development
Command ===> _____

          Package: STEV000288           Status: DEV           Install date: 20161120
          Change rqst: 00000001         Location: HERE

ISPF Library:
Project . . . . ZMFSD
Group . . . . . DB2
Type . . . . . JCL
Member . . . . _____ (Blank/pattern for list; * for all members)

Other partitioned, sequential or HFS dataset:
DSN . . . . . _____ +
Org . . . . . _____ (PDS, Seq, PAN, LIB, Oth, HFS)

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

Enter "/" to select option
/ Confirm request                _ Expand HFS subdirectories
_ Lock component                 _ Display component user options
/ Extract Stored Procedure from Db2 catalog

```

Having chosen that option we are presented with a new panel from which we can stage the SP into the package (see the help panel for a description of what all the fields mean):

```

CMNSTG25                               Stage Native-SQL SP from Db2
Command ===> _____

          Package: STEV000288           Status: DEV           Install date: 20161120

Stored Procedure:
Db2 id . . . . . DSN
Location . . . . _____
Schema . . . . . ZMFSD +
Name . . . . . LONG_NAME_SP_NUMBER_3 +
Version . . . . _____ +
Version Ind. . . _____

Component:
Name . . . . . LONGSP03
Library type . NSQ

Enter "/" to select option
/ Add package information to component
_ Lock component in package

```

The software goes to the Db2 catalog, extracts the SP code, and attempts to stage it into the package. We get an overlay warning.



```

CMNSTG25                               Stage Native-SQL SP from Db2
Command ==> _____

      Package: STEV000288           Status: DEV           Install date: 20161120
+-----+-----+-----+-----+-----+-----+-----+-----+
Stored | CMNSTGWP                               |
Db2 | Command ==> _____ |
LocSch |                                         |
SchNam | Staging Member:                       |
NamVer | LONGSP03                                 |
Ver | Will overlay ZMFSD           version. |
      |                                         |
Compon |                                         |
Nam |                                         |
Lib +-----+-----+-----+-----+-----+

Enter "/" to select option
 / Add package information to component
_ Lock component in package

```

On confirmation of the overlay the checkin service runs and stages the component to the package, we get the following message from checkin:

```

CMN408I - STEV000288 Component LONGSP03.NSQ ACTIVATED 20160222 112807. CN(INTERNAL)
***

```

If we look at the component in the package we can see the SP definition as *it was delivered to Db2 by Data Studio* (note that we are now ALTERing the proc).

The first three (optional) comment lines have been added by ZMF.

```

-- +ZMF+-----+-----+-----+-----+-----+-----+-----+-----+
-- | Pkg: STEV000288 Ltp: NSQ Uid: ZMFSD Time: 2016/02/22-11.28.07 |
-- +ZMF+-----+-----+-----+-----+-----+-----+-----+-----+
ALTER PROCEDURE ZMFSD.LONG_NAME_SP_NUMBER_3
ADD VERSION V2 ( )
ISOLATION LEVEL CS
RESULT SETS 1
LANGUAGE SQL
P1: BEGIN
-- #####
-- # Returns all tables created by ZMFV2
-- #####
-- Declare cursor
DECLARE cursor1 CURSOR WITH RETURN FOR

SELECT NAME FROM SYSIBM.SYSTABLES WHERE CREATOR = 'ZMFV2'
ORDER BY NAME;

-- Cursor left open for client application
OPEN cursor1;
END P1

```

The component is now active in the package.

We first promote it to the LOCALVER site, first promotion level. This delivers the component to ZMFSD.VUNIT.NSQ and, as a result of the active library definition and the fact that this libtype is Db2 indicated with a Db2 subtype of N, we take a number of options in the promotion skeleton.

CMNDB2DD is informed that the target logical subsystem is using Db2 versioning for Native SQL SPs, it issues (amongst others) the following messages:

```
CMNDD041I          Not a CREATE, autodrop will do nothing at this time.
CMNDD037I          Stored Procedure version information has been written to the
                   VERSION DDname.
CMNDD001I          Templated SQL sentence extracted from member LONGSP03 :
-- +ZMF+-----+
-- | Pkg: STEV000288 Ltp: NSQ Uid: ZMFSD Time: 2016/02/22-11.28.07 |
-- +ZMF+-----+
ALTER PROCEDURE UNIT .LONG_NAME_SP_NUMBER_3
ADD VERSION V2 ( )
  ISOLATION LEVEL CS
  RESULT SETS 1
  LANGUAGE SQL
P1: BEGIN
-- #####
-- # Returns all tables created by ZMFV2
-- #####
-- Declare cursor
DECLARE cursor1 CURSOR WITH RETURN FOR

  SELECT NAME FROM SYSIBM.SYSTABLES WHERE CREATOR = 'ZMFV2'
  ORDER BY NAME;
-- Cursor left open for client application
OPEN cursor1;
END P1
CMNDD002I          Sentence processed successfully.
CMNDD003I          Work committed
```

We can see that the schema for the SP has been templated using the target logical subsystem settings (i.e. it has been replaced by UNIT), and the resulting definition has been presented to Db2 and processed successfully. Also, message CMNDD037I indicates that version information has been written to the VERSION DDname. This information is passed on to a subsequent job (submitted via internal reader from the promotion job) which is (optionally) held.

When it runs that job uses the version information passed to it to activate the new SP version delivered by the promote action:

```
CMNAV003I          Statement generated for SP activation:
ALTER PROCEDURE UNIT.LONG_NAME_SP_NUMBER_3 ACTIVATE VERSION V2
CMNAV004I          Version activation completed successfully
```

As part of this activation process information about the currently active version is written to the local CMNZMF.CMNDB2\_ATTRIBS table (for use by any future demotion). So, at this point in time, the active version is V2 and the prior active version has been recorded as V1.

Moving on to the promotion to remote site REMOTVER and the first level defined there. This delivers the component to ZMFSD.VQA1.NSQ and, as a result of the active library definition (which targets a connector, see above) and the fact that this libtype is Db2 indicated with a Db2 subtype of N, the promotion skeleton is driven to use the BIND DEPLOY mechanism for distributing the SP.

CMNDB2DD is supplied a series of settings from the source and target logical subsystems as identified by the target connector from the active library definition. As a result the following BIND DEPLOY command is built. It is routed back to the source Db2 subsystem via a remote call to the IBM supplied stored procedure ADMIN\_COMMAND\_DSN. The location identified for the source logical subsystem is used to do this, in this case the remote call is to DB2V11.ADMIN\_COMMAND\_DSN and that call is presented with the following command text. Note that the location for the deployment (S0W1DSN1 in this case) is determined by CMNDB2DD at run time. CMNDB2DD is actually running at the target, it sends the deploy command back to the source using the remote call.

#### Note

To be clear, there is no presentation of SQL/DDDL to Db2 at the target location. The distribution of the Native SQL SP is performed by the BIND DEPLOY command.

```
CMNDD045I          BIND DEPLOY processing requested, command(s) will be sent to
                   location: DB2V11
BIND PACKAGE(S0W1DSN1.QA) +
  DEPLOY(UNIT.LONG_NAME_SP_NUMBER_3) +
  COPYVER(V2) +
  OWNER(SERD) +
  QUALIFIER(QA) +
  ACTION(REPLACE)

DSNT232I -DSN0 SUCCESSFUL BIND FOR
          PACKAGE = S0W1DSN1.QA.LONG_NAME_SP_NUMBER_3.(V2)
```

As well as issuing the BIND DEPLOY we also generate version information and submit the 'activation' job as before. When it runs it activates this newly deployed version of the SP:

```
CMNAV003I          Statement generated for SP activation:
ALTER PROCEDURE QA.LONG_NAME_SP_NUMBER_3 ACTIVATE VERSION V2
CMNAV004I          Version activation completed successfully
```

What happens on a demote? Here we have requested a demote for the component we just promoted to site REMOTVER. Again, there is no presentation of SQL/DDDL to Db2 but also, no BIND DEPLOY either (as BIND DEPLOY is all about delivering an SP to another location, not removing it). However, version information is written (using a different transaction code, one of which indicates a demote) and the activation job is submitted.

```

CMNDD041I      Not a CREATE, autodrop will do nothing at this time.
CMNDD037I      Stored Procedure version information has been written to the
                VERSION DName.

```

When it runs it uses the information in the CMNZMF.CMNDB2\_ATTRIBS table to decide which version of the SP to re-instate. It also drops the version of the SP that was just promoted:

```

CMNAV003I      Statement generated for SP activation:
                ALTER PROCEDURE QA.LONG_NAME_SP_NUMBER_3 ACTIVATE VERSION V1

CMNAV004I      Version activation completed successfully

CMNAV016I      Statement generated to drop the demoted version of this SP:
                ALTER PROCEDURE QA.LONG_NAME_SP_NUMBER_3 DROP VERSION V2

CMNAV018I      Version dropped successfully.

```

Moving on to the install, we have an install to both a local site, U810DP, and a remote site, U810P. Both are set up to be delivered via BIND DEPLOY and, in the CMN21 job, we see for U810DP the following:

```

CMNDD045I BIND DEPLOY processing requested, command(s) will be sent to
          location: DB2V11

BIND PACKAGE(DB2V11.PROD) +
  DEPLOY(UNIT.LONG_NAME_SP_NUMBER_3) +
  COPYVER(V2) +
  OWNER(SERD) +
  QUALIFIER(PROD) +
  ACTION(REPLACE)

DSNT254I -DSN0 DSNTBCM2 BIND OPTIONS FOR
          PACKAGE = DB2V11.PROD.LONG_NAME_SP_NUMBER_3.(V2)
          ACTION      ADD
          OWNER       SERD
          QUALIFIER   PROD
          VALIDATE    RUN
          EXPLAIN     NO
          ISOLATION   CS
          RELEASE     COMMIT
          COPY
          APREUSE
          APCOMPARE
          APRETAINDUP
          BUSTIMESENSITIVE YES
          SYSTIMESENSITIVE YES
          ARCHIVESENSITIVE YES
          APPLCOMPAT V11R1

```

```

DSNT255I  -DSN0 DSNTBCM2 BIND OPTIONS FOR
          PACKAGE = DB2V11.PROD.LONG_NAME_SP_NUMBER_3.(V2)
          SQLERROR      NOPACKAGE
          CURRENTDATA   NO
          DEGREE        1
          DYNAMICRULES  RUN
          DEFER
          REOPT          NONE
          KEEP DYNAMIC  NO
          IMMEDWRITE    NO
          DBPROTOCOL    DRDA
          OPTHINT
          ENCODING       EBCDIC(01047)
          PLANMGMT      OFF
          PLANMGMTSCOPE STATIC
          CONCURRENTACCESSRESOLUTION
          EXTENDEDINDICATOR
          PATH
DSNT232I  -DSN0 SUCCESSFUL BIND FOR
          PACKAGE = DB2V11.PROD.LONG_NAME_SP_NUMBER_3.(V2)

```

Notice that the messages fed back by the deployment (actually, from the deployment via the ADMIN\_COMMAND\_DSN SP) are much more verbose when the deployment is from/to the same Db2 subsystem (DSN in this case). We also see the activation job submitted:

```

CMNAV003I Statement generated for SP activation:
          ALTER PROCEDURE PROD.LONG_NAME_SP_NUMBER_3 ACTIVATE VERSION V2

CMNAV004I Version activation completed successfully

```

The install to the U810P site proceeds along similar lines:

```

CMNDD045I BIND DEPLOY processing requested, command(s) will be sent to
          location: DB2V11

BIND PACKAGE(S0W1DSN1.PROD) +
          DEPLOY(UNIT.LONG_NAME_SP_NUMBER_3) +
          COPYVER(V2) +
          OWNER(SERD) +
          QUALIFIER(PROD) +
          ACTION(REPLACE)

DSNT232I  -DSN0 SUCCESSFUL BIND FOR
          PACKAGE = S0W1DSN1.PROD.LONG_NAME_SP_NUMBER_3.(V2)

CMNAV003I          Statement generated for SP activation:
          ALTER PROCEDURE PROD.LONG_NAME_SP_NUMBER_3 ACTIVATE VERSION V2

CMNAV004I Version activation completed successfully

```

# Support Use of zFS File Type for SP Components

## Overview

Native SQL Db2 stored procedures can have names up to 128 bytes in length and can be case sensitive. PDS member names are restricted to a maximum length of 8 bytes, which are not case sensitive.

To remove the restriction of PDS member names on the names of Native SQL stored procedures, ZMF allows stored procedures with long names to be stored and managed in ZMF as zFS-based library types.

## Component Design

The following aspects of component design enable ZMF to store and manage Native SQL stored procedures as zFS-based library types.

## ZMF Administrative Functions

You define a zFS libtype with a Db2 subtype of N. Follow the instructions in the *ChangeMan ZMF Administrator's Guide* to copy global library type definitions into the **application - Library Types Part 1 of 2 (CMNCLLT0)** panel. See the Db2 library type NSZ in the following examples:

```
CMNCLLT0          STEV - Library Types Part 1 of 2          Row 1 to 28 of 28
Command ==>> _____ Scroll ==>> CSR

  Lib
  type Description          Order Lke Seq Defer Target Sel
  _____ +
___ NSZ Native SQL stored procs hosted by zFS    0 P      Y      D
```

```
CMNDGLT0          STEV - Db2 Library Types          Row 11 to 11 of 11
Command ==>> _____ Scroll ==>> CSR

Lib
type Description          Sub End SQL
_____ type sentence
NSZ Native SQL stored procs hosted by zFS    N    @
```

And define the target directories as active:

```

                                Db2 Active Library/Directory List                                Row 8 to 28 of 38
Command ==> _____ Scroll ==> CSR

Logical
name   Type   Db2 active library or directory name
PRODLCL1  S     /cmndev/STEV/U900DP/Prod00/NSZ      +
PRODRMT1  S     /cmndev/STEV/U900P/Prod00/NSZ      +
UNIT1     S     /cmndev/STEV/promo10/NSZ +

```

The usual invocation of CMNDB2DD to process these components will be generated for the various actions, for example, promote/demote, install/backout, and so on.

### CMNDB2DD Program Processing

This program (member CMNDB2DD of the CMNZMF.LOAD distribution library) detects whether the SQLIN and STGLIB input DDnames point to a library or a zFS path name. Appropriate I/O routines are used to access the stored procedure definition.

If PASSTHRU=YES is selected, the output (templated) procedure definition is written either to a library member or a to a zFS file depending on what is allocated to the xxxxOUT DD statement.

A synonym for the MBR= sysin parameter is CMP=. (They mean the same thing: CMP=longNameStoredProcedure; or MBR=longNameStoredProcedure).

The data interface to the High-Level Language (HLL) exit includes a long component name (Language Environment [LE] field name is DB2DCMPL, REXX variable name is 'longComponent'). The original short name fields remain in place (DB2DCMPN and REXX 'component'). The long format name is always filled in. The short format name is only filled in if the component name is 8 or fewer bytes in length.

### Skeletons

The following skeletons use PATH= and PATHOPTS DD parameters when the indicated library type is a zFS libtype.

- CMN\$\$PSQ (local promote/demote)
- CMN\$\$RSQ (remote promote/demote)
- CMN\$\$SQL (install/backout)

### CMNSTGER Processing

The ISPF function program CMNSTGER, which allows you to stage a component directly from Db2 into a package, has been changed to handle a target zFS libtype, as shown in the Stage Native SQL SP from Db2 (CMNSTG25) panel in the following example:

```

CMNSTG25                               Stage Native SQL SP from Db2
Command ==>

          Package: STEV001561           Status: DEV           Install date: 20221111

Stored Procedure:
  Db2 id . . . . . D10L
  Location . . . .
  Schema . . . . . WSER58
  Name . . . . . "zFSnativeSQLstoredProcedure001"
  Version . . . .
  Version Ind . .

Component:
  Name . . . . .
  Library type. . NSZ

Enter "/" to select option
 / Add package information to component
 _ Lock component in package
 / Mixed Case

```

The following messages are displayed when panel content is processed:

```

CMN408I - STEV001561 Component zFSnativeSQLstoredProcedure001.NSZ
          ACTIVATED 20200522 085455. CN(INTERNAL)

***
and:

CMN2575I - zFSnativeSQLstoredProcedure001 component staged from D10L

```

If the component name is missing, the same name as the stored procedure name is used but with any double or single quotes removed.

### CMNVCOMP Component Checkin Service

This service also allows for the checkin of long name stored procedure components direct from Db2 into a zFS library type. The double quotes on the stored procedure name are not actually necessary: both the ISPF client and the service itself will strip both double and single quotes from stored procedure names before using them. The quotes are tolerated because Db2 requires them to avoid folding everything to upper case. We rely on the MIXCASE variable in the ISPF client and just take names as is (no folding) in the service.



```

<?xml version="1.0"?>
<service name="CMPONENT">
  <scope name="SERVICE">
    <message name="CHECKIN">
      <header>
        <subsys>L</subsys>
        <product>CMN</product>
      </header>
      <request>
        <package>STEV001561</package>
        <component>"zFSnativeSQLstoredProcedure001"</component>
        <componentType>NSZ</componentType>
        <chkInSourceLocation>D</chkInSourceLocation>
        <sourceDb2Subsys>D10L</sourceDb2Subsys>
        <sourceDb2Schema>WSER58</sourceDb2Schema>
        <addZmfInfoToDb2Object>Y</addZmfInfoToDb2Object>
        <listCount>00001</listCount>
        <targetComponent>zFSnativeSQLstoredProcedure001</targetComponent>
      </request>
    </message>
  </scope>
</service>

```

## Backward Compatibility

All existing PDS member based facilities are unchanged.

## Installation and Configuration

No environmental changes are needed to implement this enhancement to stored procedure names. You need only define one or more zFS library types to be Db2 enabled with a Db2 subtype of N (native SQL stored procedure).

# 14. Glossary

---

## Active Libraries

An Active library, as used in this document, is a promotion, production, or baseline library that contains Db2 components. The library can contain Db2 program load modules, BIND commands, or stored procedures. When a change to an active library is detected, either BIND processing or stored procedure processing is included in the promotion or installation process JCL.

## DBRM

DBRM is a Db2 Data Base Request Module. A DBRM is the output of the Db2 precompiler after it has processed a source module containing SQL statements.

## Rebind

This manual frequently refers to the process of rebinding plans. In fact, plans processed by ChangeMan are actually not rebound, but bound with the REPLACE option. In Db2, the REBIND command does not use information external to the catalog, such as a DBRM in a PDS. Instead, the REBIND command used the SYSIBM.SYSDBRM and SYSIBM.SYSPLAN tables.

BIND with the REPLACE option, however, uses information in DBRM PDSs to replace a plan in the catalog. Since ChangeMan is processing change SRC components, there is a high probability that the related DBRMs have new time stamps, and need to be updated in Db2 plans.

## SPUFI

SPUFI is an acronym for SQL Processor Using File Input. SPUFI can be used to directly enter SQL statements without having to write a program to process those statements.

## SQL

SQL is Structured Query Language, the data manipulation language for Db2.

## Versioning

Versioning in Db2 is the ability to run multiple versions of a program. Multiple versions of a program can exist with the same Collection Id and NAME combination. The sequence of concatenation of load libraries determines which program executes and Db2 will find a matching package ready for execution amongst the many that are available.

# 15. Legal Notice

---

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

© Copyright 2023 Micro Focus or one of its affiliates.

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

## Third-Party Notices

---

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

## Specific notices

---

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