BMC Infrastructure Components
Administration Guide

Supporting

Version 10.1.00 of DB2 Component Services (DBC)
Version 10.1.00 of DB2 Product Configuration (LGC)
Version 10.3.00 of Next Generation Logger (NGL)

December 2016
Contacting BMC Software

Several methods are available for contacting BMC Software.

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

United States and Canada

<table>
<thead>
<tr>
<th>Address</th>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC SOFTWARE INC 2103 CITYWEST BLVD HOUSTON TX 77042-2827 USA</td>
<td>1 713 918 8800 or 1 800 841 2031</td>
<td>1 713 918 8000</td>
</tr>
</tbody>
</table>

Outside United States and Canada

<table>
<thead>
<tr>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>+01 713 918 8800</td>
<td>+01 713 918 8000</td>
</tr>
</tbody>
</table>

© Copyright 2013-2016 BMC Software, Inc.

BMC, BMC Software, the BMC logo, the BMC Software logo, and other BMC marks are the exclusive properties of BMC Software, Inc. and are registered or may be registered in the U.S. and in other countries. BladeLogic and other BladeLogic marks are the exclusive properties of BladeLogic, Inc. and are registered or may be registered in the U.S. and in other countries. All other trademarks or registered trademarks are the property of their respective owners.

Explorer is a trademark or registered trademark of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

DB2, IBM, IBM z, IMS, MVS, RACF, WebSphere, z/OS, and zEnterprise are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

UNIX is the registered trademark of The Open Group in the US and other countries.

The information included in this documentation is the proprietary and confidential information of BMC Software, Inc., its affiliates, or licensors. Your use of this information is subject to the terms and conditions of the applicable End User License agreement for the product and to the proprietary and restricted rights notices included in the product documentation.

Restricted rights legend

U.S. Government Restricted Rights to Computer Software. UNPUBLISHED—RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure of any data and computer software by the U.S. Government is subject to restrictions, as applicable, set forth in FAR Section 52.227-14, DFARS 252.227-7013, DFARS 252.227-7014, DFARS 252.227-7015, and DFARS 252.227-7025, as amended from time to time. Contractor/Manufacturer is BMC SOFTWARE INC, 2103 CITYWEST BLVD, HOUSTON TX 77042-2827, USA. Any contract notices should be sent to this address.
Customer support

Support website
You can obtain technical support from BMC 24 hours a day, 7 days a week at http://www.bmc.com/support. From this website, you can:

- Read overviews about support services and programs that BMC offers
- Find the most current information about BMC products
- Search a database for problems similar to yours and possible solutions
- Order or download product documentation
- Download products and maintenance
- Report a problem or ask a question
- Subscribe to receive proactive e-mail alerts
- Find worldwide BMC support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

Support by telephone or e-mail
In the United States and Canada, if you need technical support and do not have access to the web, call 1 800 537 1813 or send an e-mail message to customer_support@bmc.com. (In the subject line, enter SupID:yourSupportContractID, such as SupID:12345). Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC
Have the following information available so that Customer Support can begin working on your issue immediately:

- Product information
  - Product name
  - Product version (release number)
  - License number and password (trial or permanent)
- Operating system and environment information
  - Machine type
  - Operating system type, version, and service pack or other maintenance level such as PUT or PTF
  - System hardware configuration
  - Serial numbers
  - Related software (database, application, and communication) including type, version, and service pack or maintenance level
- Sequence of events leading to the problem
- Commands and options that you used
- Messages received (and the time and date that you received them)
  - Product error messages
  - Messages from the operating system
  - Messages from related software
License key and password information

If you have questions about your license key or password, contact Customer Support through one of the following methods:

■ Send an e-mail message to customer_support@bmc.com. (In the Subject line, enter SupID:yourSupportContractID, such as SupID:12345.)

■ In the United States and Canada, call 1 800 537 1813. Outside the United States and Canada, contact your local support center for assistance.

## Contents

### About this book
- Related publications ......................................................... 9
- Syntax statements ............................................................. 10
- Conventions .................................................................................. 11

### Chapter 1  BMC infrastructure components 13
- Overview of DB2 Component Services (DBC) .......................... 14
- Overview of DB2 Product Configuration (LGC) ...................... 15
- Overview of the Next Generation Logger (NGL) ...................... 16
- Overview of the Runtime Component System (RTCS) ............ 16
- Required authorizations for user IDs ........................................ 17

### Chapter 2  Working with DB2 Component Services (DBC) 21
- Working with the DBC subsystem ........................................... 21
  - Started task for the DBC subsystem .................................... 21
  - DBC startup parameters ...................................................... 23
  - Stopping the DBC subsystem .............................................. 32
  - DBC commands ................................................................... 33
- DBC security ............................................................................. 36
  - Controlling access to DBC resources .................................. 37
  - Configuring RACF security for the DBC subsystem ............ 38
- DBC security parameters ......................................................... 41

### Chapter 3  Working with DB2 Product Configuration (LGC) 47
- Overview of the LGC interface ................................................. 48
  - Expandable sections ......................................................... 48
  - Drop-down menus ............................................................ 50
  - Action list ........................................................................... 51
  - Online Help ......................................................................... 51
- Working with option sets ......................................................... 51
  - Accessing the online interface .......................................... 52
  - Creating a new option set .................................................. 53
  - Editing an option set ......................................................... 54
  - Renaming an option set ..................................................... 57
  - Deleting an option set ......................................................... 58
  - Filtering option sets ............................................................ 58
About this book

This book contains information about managing and deploying infrastructure components that certain BMC mainframe products use. This book is intended for system administrators.

The infrastructure components described in this book include:

- DB2 Component Services (DBC)
- DB2 Product Configuration (LGC)
- Next Generation Logger (NGL)
- Runtime Component System (RTCS)

**Note**
For information about infrastructure that is exclusive to MainView, see your MainView documentation.

Like most BMC documentation, this book is available in printed and online formats. To request printed books or to view online books and notices (such as release notes and technical bulletins), see the support website at [http://www.bmc.com/support](http://www.bmc.com/support).

**Note**
Online books are formatted as PDF or HTML files. To view, print, or copy PDF books, use the free Adobe Reader from Adobe Systems. If your product installation does not install the reader, you can obtain the reader at [http://www.adobe.com](http://www.adobe.com).

The software also offers online Help. To access Help, press F1 within any product or click the Help button in graphical user interfaces (GUIs).

Related publications

From the BMC Support Central website, you can use the following methods to access related publications that support your product or solution:

View Quick Course videos (short overviews of selected product concepts, tasks, or features), which are available from the following locations:

— Documentation Center (primary center and secured center)
— Support Central (at http://www.bmc.com/support/mainframe-demonstrations)
— BMC Mainframe YouTube channel (https://www.youtube.com/user/BMCSoftwareMainframe)


Products with online interfaces also offer online Help via the F1 key or, for graphical user interfaces (GUIs), via a Help button.

Tip
If you prefer hardcopy documentation, you can order it from your BMC sales representative or from Support Central. Also, from Support Central you can subscribe to receive proactive e-mail alerts when BMC issues notices.

Syntax statements

This topic explains conventions for showing syntax statements.

A sample statement follows:

COMMAND KEYWORD1 [KEYWORD2 | KEYWORD3] KEYWORD4={YES | NO} fileName...

The following table explains conventions for syntax statements and provides examples:
Items in italic type represent variables that you must replace with a name or value. If a variable is represented by two or more words, initial capitals distinguish the second and subsequent words.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Example</th>
</tr>
</thead>
</table>
| Items in italic type represent variables that you must replace with a name or value. If a variable is represented by two or more words, initial capitals distinguish the second and subsequent words. | alias
databaseDirectory
serverHostName |
| Brackets indicate optional items. Do not type the brackets when you enter the option. A comma means that you can choose one or more of the listed options. You must use a comma to separate the options if you choose more than one option. | [tableName, columnName, field]
[-full, -incremental, -level] |
| Braces indicate that at least one of the enclosed items is required. Do not type the braces when you enter the item. | {DBDName | tableName}
UNLOAD device={disk | tape, fileName | deviceName}
{-a | -c} |
| A vertical bar means that you can choose only one of the listed items. In the example, you would choose either commit or cancel. | {commit | cancel} |
| An ellipsis indicates that you can repeat the previous item or items as many times as necessary. | columnName... |

**Conventions**

This document uses the following special conventions:

- All syntax, operating system terms, and literal examples are presented in this typeface.

- Variable text in path names, system messages, or syntax is displayed in italic text: `testsys/instance(fileName)

- Menu sequences use a symbol to convey the sequence. For example, **Actions => Create Test** instructs you to choose the **Create Test** command from the **Actions** menu.
BMC infrastructure components

This publication introduces infrastructure components that are used by many BMC mainframe products, and explains how to deploy and manage them.

**Note**

For information about infrastructure that is exclusive to MainView, see your MainView documentation.

The User Interface Middleware (UIM) and EXTENDED BUFFER MANAGER (XBM) are additional components that are also configured in the installation dialog under Infrastructure. For more details, see *BMC Products and Solutions for DB2 Customization Guide* and *EXTENDED BUFFER MANAGER and SNAPSHOT UPGRADE FEATURE User Guide* respectively.

<table>
<thead>
<tr>
<th>Component</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| DB2 Component Services (DBC) | DBC provides a persistent z/OS subsystem address space into which BMC products can dynamically initialize their own product services. The following components also require the DBC subsystem:  
- The Data Collector for System and SQL Performance products  
- BMC Workbench for DB2 (also called the GUD component)  
- LGC and NGL components  
- BMC Subsystem Optimizer for zEnterprise |
| DB2 Product Configuration (LGC) | The LGC technology separates product or solution installation from configuration. Through its online interface, LGC simplifies configuration and deployment by setting default option values for you. |
| Next Generation Logger (NGL) | NGL is a logging facility that logs and retrieves data based on application-defined keys and a time span. NGL runs as a service within the DBC subsystem and relies on the Registry Component of RTCS to permanently store information used to manage the logged application data. |
| Runtime Component System (RTCS) | RTCS runs as a started task and provides programming services to various BMC mainframe products. RTCS is designed for continuous operation and seldom, if ever, needs to be stopped. |

Table 2 on page 14 indicates the products that use each component.
Table 2: Products that use the infrastructure components

<table>
<thead>
<tr>
<th>Product</th>
<th>DBC</th>
<th>LGC</th>
<th>NGL</th>
<th>RTCS a</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPTUNE for DB2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MainView for DB2 - Data Collector</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MainView for Java Environments</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MainView for MQ</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MainView Infrastructure</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MainView Transaction Analyzer</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PACLOG for DB2</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pool Advisor for DB2</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BMC Recovery Management for DB2</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>RECOVERY MANAGER for DB2</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQL Performance for DB2</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BMC Subsystem Optimizer for zEnterprise (Subzero)</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BMC System Performance for DB2</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>BMC Workbench for DB2</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

a Additional products also use RTCS. For more information about RTCS, see BMC Runtime Component System Configuration and Administration Guide. The BMC Infrastructure Components Administration Guide describes RTCS only as it interacts with the other infrastructure components.

Typically, these components are shared across the BMC products with one instance per LPAR and are upgraded only through maintenance releases.

Overview of DB2 Component Services (DBC)

The DB2 Component Services (DBC) technology provides a persistent z/OS subsystem address space into which enabled BMC products can dynamically initialize their own product services.

DBC uses a variety of techniques to accommodate dynamic initializations:

- Through an XML messaging protocol, DBC provides a non-authorized, loosely coupled, sysplex-enabled communication channel to product services.
DBC hosts common services for IBM DB2 subsystem discovery and command execution.

DBC allows BMC products to define operator commands, and to subscribe to and publish user events dynamically.

All product services that are hosted within the DBC infrastructure inherit a Security Access Facility (SAF) interface to ensure compliance with the relevant site's security requirements.

The FMID that corresponds to DBC technology is ZDBCxxx, where xxx represents release number information.

**Note**
For information about maintaining and using the DBC, see “Working with DB2 Component Services (DBC)” on page 21. For information about using the DBC with a particular product, see that product's documentation.

---

## Overview of DB2 Product Configuration (LGC)

DB2 Product Configuration (LGC) technology separates product or solution installation from configuration.

Through its online interface, LGC simplifies configuration. You can accept the default option values or make changes to them, if needed.

LGC panels simplify navigation by allowing you to expand or contract sections as needed. You can link to LGC from within your product or solution to maintain a consistent look and feel, and to retain your changes from version to version.

LGC operates as an agent within the DBC subsystem. The LGC agent uses the Registry Component of the Runtime Component System (RTCS) to access the RTCS Subsystem system registry or, optionally, a private RTCS-managed registry. LGC stores the option sets for many BMC products in a datastore (a collection of XML documents), which is stored in one of the RTCS-managed registry instances.

LGC can use a private, shared RTCS-managed registry, which uses the MVS cross-system coupling facility (XCF) to communicate with other LGC agents in the sysplex defined to the same DBC group. A private registry instance in one and only one LGC agent at a time owns (dynamically allocates and performs all I/O) the backing VLDS for the RTCS-managed registry which contains the LGC datastore. If the owning agent is stopped, an RTCS registry instance in another LGC agent in the same DBC group assumes ownership of the registry after first dynamically allocating its backing VLDS. If LGC is using the RTCS Subsystem system registry for the LGC
datastore, the RTCS Subsystem address spaces manage their common registry instances and the single sysplex-shared backing VLDS.

The FMID that corresponds to the LGC technology is ZLGCxxx, where xxx represents release number information.

Note
For more information about using LGC with a particular product, see “Working with DB2 Product Configuration (LGC)” on page 47.

Overview of the Next Generation Logger (NGL)

Next Generation Logger (NGL) technology allocates, initializes, and manages log files. NGL minimizes the cost of logging and the potential for resource contention.

The Next Generation Logger operates as agents within the DBC subsystems. Multiple NGL instances can be active that are uniquely identified by a product instance identifier (PIID). For example, the MainView Transaction Analyzer product might use NGL1 for the PIID, while the System and SQL Performance products use NGL2.

The FMID that corresponds to the NGL technology is ZNGLxxx, where xxx represents release number information.

Note
For information about maintaining and supporting NGL, see “Working with the Next Generation Logger (NGL)” on page 93. For information about using NGL with a particular product, see that product's documentation.

Overview of the Runtime Component System (RTCS)

The Runtime Component System (RTCS) technology provides programming services to BMC mainframe products. By supporting both component-based and traditional procedural programming, RTCS uses the latest z/OS system facilities to take advantage of modern enterprise servers.
RTCS simplifies product installation and configuration. Other products use the services and components made available by RTCS to support additional functions, replace or update functions, or extend existing functions.

The FMIDs that correspond to the RTCS technology are ZOSZxxx and LOSZxxx, where xxx represents release number information.

*Note*
For more information about supporting and maintaining RTCS, see the *BMC Runtime Component System Configuration and Administration Guide*.

### Required authorizations for user IDs

Multiple user IDs are associated with installing, configuring, and using the common components with BMC products. This section describes the permissions and security settings for these IDs:

- “Installation user ID” on page 17
- “DBC started task user ID” on page 18
- “NGLARCH started task user ID” on page 19
- “User ID” on page 19

#### Installation user ID

The user ID of the installer must have the following permissions and security settings:

- ALTER authority for the following data sets:
  - BMC Installation System installation data sets
  - SMP/E global, target, and distribution data sets
  - Runtime data sets
  - User data sets

- READ authority for the IBM Resource Access Control Facility (RACF) FACILITY class for the following resources:
  - BMC.DBC.*
  - BMC.DPR.*
— BMC.LGC.* (if LGC is installed)
— BMC.NGL.* (if NGL is installed)

■ USS SUPERUSER access

**DBC started task user ID**

The started task for the DBC must have the following permissions and security. For more information about DBC, see “Working with DB2 Component Services (DBC)” on page 21.

■ DBC must meet the following UNIX requirements:
  — Write and execute access to the `/tmp` directory.
  — Update access to the FSACCESS (UNIX file system access check) resource class.

■ DBC must be authorized to create an Extended MCS Console.

■ READ authority for the RACF FACILITY class for the following resources:
  — BMC.DBC.*
  — BMC.DPR.*
  — BMC.LGC.* (if LGC is installed)
  — BMC.NGL.* (if NGL is installed)

■ ALTER authority for the user data sets (that is, DBCREPOS and LOGSET files)

■ READ and WRITE authority for the:
  — LGC private registry data set (if LGC is installed)
  — NGL private registry data set (if NGL is installed)

■ An OMVS segment defined in the IBM RACF (normal user) security product or an equivalent security product

■ When using APPTUNE object data collection, READ authority for:
  — DB2 Version 9 subsystems data sets:
    — `db2cat.DSNDBD.DSNDB06.SYSDATABASE.I0001.A001`
    — `db2cat.DSNDBD.DSNDB06.SYSUSER.I0001.A001`
  — DB2 Version 10 and later subsystems data sets:
    — `db2cat.DSNDBD.DSNDB06.SYSTSTAB.I0001.A001`
    — `db2cat.DSNDBD.DSNDB06.SYSTSIXS.I0001.A001`
    — `db2cat.DSNDBD.DSNDB06.SYSUSER.I0001.A001`
When using Pool Advisor, READ authority for DB2 Version 10 and later subsystems data sets:

- `db2cat.DSNDBD.DSNDB06.SYSTSDBA.I0001.A001`
- `db2cat.DSNDBD.DSNDB06.SYSTSTAB.I0001.A001`
- `db2cat.DSNDBD.DSNDB06.SYSTSSTP.I0001.A001`
- `db2cat.DSNDBD.DSNDB06.SYSTSIXS.I0001.A001`

READ authority for System Authorization Facility (SAF) class DSNR for:

- `db2ssid.BATCH`
- `db2ssid.RRSAF`

**NGLARCH started task user ID**

The started task for the NGL must have the following permissions and security:

- ALTER authority for the HLQ for the user data sets (that is, LOGSET files)
- READ and WRITE authority for the NGL private registry data set (*if NGL is installed*)
- An OMVS segment defined in IBM RACF (normal user) or the equivalent in your security system

**User ID**

To use interface components of the products, the user ID must have:

- READ authority for the runtime data sets
- READ authority for the RACF FACILITY class for the following resources:
  - `hlq.DBC.*`
  - `hlq.DPR.*`
- An OMVS segment defined in the RACF (normal user) security product or an equivalent security product
- Execute access to the `/tmp` directory
- Any User ID that issues operator commands to the DBC must have READ authority for the RACF FACILITY class for the following resource: `hlq.lpar.dbcgroup.prodCode.command.PF`

The variables are defined as follows:
— *hlq* is the high-level qualifier of the resource name. The HLQ node defaults to BMC, but you can customize the value by using the `<HLQ>` option in the DBC SAF startup options.

— *lpar* is the MVS system name where DBC executes.

— *dbcgroup* is the name of the DBC. This name is specified either in the execution parms for the DBC started task or in the DBCPARMS input DD statement. This name is also the XCF group name for the DBC.

— *prodCode* is the BMC product code of the product for which the resource is defined. This three-character code is specified in the INITPROD command used in product initialization.

— *command* is the name of the command.

---

**Note**

If the resource rule for an operator command *does not exist* and the SAF security product returns RC=4, the operator command is allowed irrespective of the ALLOW_SAF_RC4 setting. Existing rules are then checked subject to the ALLOW_SAF_RC4 setting.

If the resource rule for an operator command *does exist*, subsequent checks for existing rules are bypassed.

You can use a wildcard for any of these nodes when you define a resource rule.
Working with DB2 Component Services (DBC)

This chapter explains how to start and manage the DBC subsystem and how to manage DBC security.

Working with the DBC subsystem

To use the DBC technology, you must start and manage a DBC subsystem.

This section explains how to:

- Start a DBC subsystem
- Specify DBCPARMS parameters
- Stop a DBC subsystem

**Note**
If you are using a version of IBM z/OS earlier than Version 1 Release 12, ensure that the AF_UNIX MAXSOCKETS value in BPXPRMxx is set to a value of at least 1000 before starting the DBC subsystem. For more information about the MAXSOCKETS value, refer to IBM documentation.

Started task for the DBC subsystem

Normally, you start the DBC subsystem as a z/OS started task.
**WARNING**

When performing an IPL, wait until after OMVS initialization has completed to start the DBC. OMVS is the task that initializes z/OS Unix System Services (USS). If DBC detects that USS is not initialized, it waits up to 30 minutes for the initialization to complete. For every minute that it waits for USS initialization, DBC issues message BMCDBC0315W:

BMCDBC0315W 10.33.23 WAITING ON UNIX SYSTEM SERVICES

After waiting for 30 minutes for USS initialization, DBC terminates with message BMCCDB0316E

BMCCDB0316E 11.03.27 UNIX SYSTEM SERVICES NOT AVAILABLE

You should add the JCL procedure for the started task into a system procedure library.

**Note**

For testing or trial installations, you can also start the DBC subsystem as a batch job. However, the JES initiator will be busy for the life of the DBC subsystem. BMC does not recommend this approach for non-trial installations.

The following figure shows an example of the started task for DBC.

**Figure 1: JCL procedure for the DBC started task**

```bash
//*********************************************************************
//*
//* Description:
//*   BMC Software DBC subsystem JCL procedure for the started task.
//* *
//* Customization Steps:
//*   - Modify the DBC subsystem initialization parameters in the
//*     member DBC$PARM that the DBCPARMS DD statement identifies.
//*   - Modify the DBC subsystem security parameters in the member
//*     DBC$SECU that the DBCSECU DD statement identifies.
//*   - Allocate Registry data set for products running under DBC
//*     Sample define:
//*       DEFINE CLUSTER (NAME(BMCDBC.ssid.REGISTRY) -
//*        LINEAR CYL(25 10) SHAREOPTIONS(1,3) STORCLAS(xxxxxx))
//*   - Add this JCL procedure to a system procedure library.
//*   - APF authorize the DBC STEPLIB data set.
//* *
//* Notes:
//*   The DBC subsystem is a long-running-service address space that
//*   normally remains active for the life of an IPL. Therefore, BMC
//*   does not recommend starting the DBC subsystem as a batch job.
//*   Doing so causes the JES initiator to be busy for the life of
//*   the DBC subsystem. If you want to run the DBC as a batch
//*   job, replace the PROC statement with a valid JCL job card.
//* *
//*********************************************************************

DBC      PROC SSID=,GRP=,REPMODE=
//DBC    EXEC PGM=DBCMAIN,REGION=0M,ACCT=&ACC,TIME=&TIM,
//         PARM='SSID=&SSID,GROUP=&G,TRACE=&T'
//STEPLIB  DD DSN=BMCPERF.TEST.BMCLINK,
//         DISP=SHR
//DBCPRINT DD SYSOUT=*,RECFM=VA
//SYSPRINT DD SYSOUT=*,RECFM=VA
```
Table 3 on page 23 describes DD statements that you define within the started task.

Table 3: DD statements for the DBC started task

<table>
<thead>
<tr>
<th>DD statement</th>
<th>Identifies the</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPLIB</td>
<td>DBC load library</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The DBC STEPLIB load library must be APF authorized.</td>
</tr>
<tr>
<td>DBCPRINT</td>
<td>Data set to dynamically write DBC messages</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Data set to write potential non-DBC product application messages</td>
</tr>
<tr>
<td>BMCPSWD</td>
<td>Data set containing the product license keys</td>
</tr>
<tr>
<td>DBCPARMS</td>
<td>DBC initialization parameters file</td>
</tr>
<tr>
<td></td>
<td>For more information, see “DBC startup parameters” on page 23.</td>
</tr>
<tr>
<td>DBCSECUR</td>
<td>Security parameters file</td>
</tr>
<tr>
<td></td>
<td>For more information, see “DBC security parameters” on page 41.</td>
</tr>
<tr>
<td>DOMTMPLT</td>
<td><em>(for System and SQL Performance products only)</em> Data set that contains advisor text</td>
</tr>
<tr>
<td>DOMPARMS</td>
<td><em>(for System and SQL Performance products only)</em> Data set that contains parameters for advisors</td>
</tr>
</tbody>
</table>

DBC startup parameters

The DBC subsystem has the following sets of logically discrete startup parameters:

- **DBCPARMS** parameters include the required DBC subsystem ID (SSID) and optional parameters, such as the XCF group name and repository data set name. You specify this general set of control parameters through the DBCPARMS DD statement. For more information, see “DBCPARMS parameters” on page 24.

- **DBCSECUR** parameters relate specifically to security customization. You identify these parameters through the DBCSECUR DD statement. DBC maintains the security parameters as a discrete set of parameters so that you can implement a
different level of data set security for these parameters, if needed. For more information, see “DBC security” on page 36.

**Note**
The only required DBC initialization parameter is the SSID; along with the XCF group name, you can also specify the SSID via the EXEC PGM=DBCMAIN,PARM='parms' JCL statement. If you specify the DBC SSID or GROUP through the JCL PARM= override statement, these values take precedence over any <SSID> or <GROUP> XML element values specified in DBCPARMS statement.

Figure 2 on page 24 shows an example of the started task JCL with parameters.

**Figure 2: Started task for DBC**

```plaintext
//DBC      PROC
//DBCEXEC  EXEC PGM=DBCMAIN,PARM='SSID=DO&SYSCLONE,GROUP=DBCGROUP,REPMODE='
//DBCxxx10 EXEC PGM=DBCMAIN,REGION=0M,ACCT=xxxx,TIME=1440,
          PARM=('SSID=&SSID,MODE=&REPMODE,GROUP=&GRP')
//STEPLIB  DD DISP=SHR,DSN=BMC.DBC.BMCLINK
//SYSPRINT DD SYSOUT=*,RECFM=VA
//DBCPARMS DD DISP=SHR,DSN=BMC.DBC.UBMCSAMP(DBC$PARM)
//DBCSECUR DD DISP=SHR,DSN=BMC.DBC.UBMCSAMP(DBC$SECU)
```

**DBCPARMS parameters**
The DBCPARMS DD JCL statement identifies the initialization parameters file for the DBC started task.

**Note**
Initialization parameters are separated from the DBC security startup parameters that you specify in the DBCSECUR DD statement. This separation allows you to manage the security parameters separately from other DBC parameters by implementing data set name security through your External Security Manager (ESM).

For more information about the started task, see “Started task for the DBC subsystem” on page 21. For more information about security parameters, see “DBC security parameters” on page 41.

**Guidelines for sharing DBCPARMS across multiple DBC subsystems**

If you want to share a single DBCPARMS parameters file across all DBC subsystems, consider the following guidelines:
The DBC SSID must be unique within an XCF group and unique on a single LPAR. You must specify the required DBC SSID by using the JCL parameter override statement (EXEC PGM=DBCMAIN,PARM='SSID=ssid').

You can use the &SYSCLONE symbolic from the IBM MVS system symbols as part of the SSID= parameter. Doing so allows you to create unique DBC SSIDs across the sysplex without having separate PROCs. You can create two-, three-, or four-character IDs by combining &SYSCLONE with other literal characters (for example, SSID=DC&SYSCLONE). To determine the value of &SYSCLONE at your site, contact your system programmer.

By default, the DBC starts an XCF group based on the name of the DBC group. You can specify the DBC group name by using the JCL parameter override statement (EXEC PGM=DBCMAIN,PARM='GROUP=group'). You can specify the XCF group name by using the <XCFGROUP> element in the DBCPARMS member.

You can specify the DBC repository data set name in the DBCPARMS parameters file by using one or more DBC system variables. Doing so ensures a unique repository data set name for each DBC subsystem instance. Table 4 on page 25 lists the DBC system variables that you can use to create a unique data set name.

Table 4: DBC system variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;SSID.</td>
<td>4-byte subsystem ID of the DBC subsystem</td>
</tr>
<tr>
<td>&amp;SMFID.</td>
<td>4-byte SMF ID of the host system</td>
</tr>
<tr>
<td>&amp;MVSNAME.</td>
<td>8-byte system name of the host system</td>
</tr>
</tbody>
</table>

For more information about the individual DBC initialization parameters, see “DBCPARMS elements” on page 26.

Structure of the XML stream

Figure 2 on page 24 shows a sample parameters file.

```xml
<DBCPARMS>
  <OPTIONS>
    <SSID>SSID</SSID>
    <GROUP>BMC\GROUP</GROUP>
    <XCFGROUP>DBC\XC\GROUP</XCFGROUP>
    <MODE>NORMAL</MODE>
    <WTouppercase>YES</WTouppercase>
  </OPTIONS>
</DBCPARMS>
```
For JCL sample files, see the SAMP data set.

**DBCPARMS elements**

*(required)* The `<DBCPARMS>` element is the root-level element in the XML stream that is used to define DBC parameters.

**Data type:** Not applicable.

**Child elements:** `<OPTIONS>`

**OPTIONS**

*(required)* The `<OPTIONS>` element contains the DBCPARMS values.

**Data type:** Not applicable.

**Parent element:** `<DBCPARMS>`

**Child elements:** `<SSID>`, `<GROUP>`, `<WTOUPPERCASE>`, `<XCFGROUP>`, `<MODE>`, `<WLM>`, and `<DPRREPOS>`

**SSID**

*(optional)* The `<SSID>` element specifies the SSID of the DBC subsystem. The SSID is a required DBC startup parameter. You must specify the SSID either through the DBCPARMS XML document or in the EXEC statement; otherwise, initialization fails.

Consider the following requirements for the value of the `<SSID>` element:

- BMC recommends that you use different names for the DBC SSID and the DBC started task.
- The SSID value must not conflict with any MVS command verbs. If it does, DBC issues an error message and terminates.
- The SSID value can contain A–Z, 0–9, #, or $. Any other characters are invalid and cause the DBC subsystem to issue an error message and terminate.

- The SSID value should not begin with a numeric character (0–9). You can use the SSID to issue system commands to the DBC subsystem; consequently, if the SSID value started with a number, the system could misinterpret the value as the short form of the REPLY system command. (For more information about issuing system commands, see “DBC commands” on page 33.)

- The SSID value should not conflict with any JES commands. This recommendation ensures that only the DBC subsystem processes system commands that you issue to the DBC subsystem by using the DBC SSID value.

  **Data type:** VARCHAR(4)
  **Parent element:** <OPTIONS>
  **Child elements:** None.

### GROUP

*(optional)* The <GROUP> element specifies the name of the DBC XCF group that relates DBC subsystems on different LPARs within the SYSPLEX.

**Note**

You can also specify the DBC XCF group through the EXEC PGM=DBCMAIN,PARM='parms' JCL statement.

Consider the following requirements for the value of the <GROUP> element:

- The GROUP value must not conflict with any MVS command verbs. If it does, DBC issues an error message and terminates.

- The GROUP value must not contain embedded blank characters.

- The GROUP value should not begin with a numeric character (0–9). You can use the group name to issue system commands to the DBC subsystem; consequently, if the group name started with a number, the system could misinterpret the value as the short form of the REPLY system command. (For more information about issuing system commands, see “DBC commands” on page 33.)

- The GROUP value should not conflict with any JES commands. This recommendation ensures that only the DBC subsystem processes system commands that you issue to the DBC subsystem by using the GROUP value.
Note
A DBC XCF group can contain only one DBC subsystem per LPAR.

The default value is DBCGROUP.

- **Data type:** VARCHAR(8)
- **Parent element:** <OPTIONS>
- **Child elements:** None.

**XCFGROUP**

*(optional)* By default, DBC uses the <GROUP> element as the name of an XCF group to join at startup. In rare situations, this behavior might not be acceptable. Use the <XCFGROUP> element to specify a different XCF group for DBC to join.

- **Data type:** VARCHAR(8)
- **Parent element:** <OPTIONS>
- **Child elements:** None.

**MODE**

MODE controls whether to initialize products stored in the repository when starting. Valid values for the <MODE> element are SAFE and NORMAL.

- **SAFE** mode tells the DBC subsystem to start without automatically initializing products stored in the repository.

- **NORMAL** mode (the default value) initializes products stored in the repository.

You can also specify the value for MODE through start arguments to DBCMAIN.

If DBC starts in SAFE mode, DBC issues the following message to the system log:

```
BMCDBC0250 ssid DBC has been requested to ignore INITPROD commands from the repository
```

If starting in default NORMAL mode, DBC issues the following message:

```
BMCDBC0234I DBC operating in NORMAL mode.
```

- **Data type:** VARCHAR(6)
- **Parent element:** <OPTIONS>
- **Child elements:** None.

**WLM**

The <WLM> element sets the subsystem type to use for workload management. The default is BMCS.
**Data type:** Not applicable.

**Parent element:** `<OPTIONS>`

**Child elements:** `<SUBSYSTYPE>`

---

**SUBSYSTYPE**

The `<SUBSYSTYPE>` element contains the name of the subsystem type.

- **Data type:** VARCHAR(8)
- **Parent element:** `<OPTIONS>`
- **Child elements:** `<SUBSYSTYPE>`

---

**WTOUPPERCASE**

*(optional)* The `<WTOUPPERCASE>` element indicates whether to convert console messages to uppercase. Printed messages remain in mixed case.

The default value is YES. Valid values are YES and NO.

- **Data type:** VARCHAR(3)
- **Parent element:** `<OPTIONS>`
- **Child elements:** None.

---

**DPRREPOS**

*(optional)* The `<DPRREPOS>` element contains the DPR repository name and allocation options. If you do not specify this element, the repository services of the DPR component are unavailable and the DBC issues a warning message during initialization; however, DBC subsystem services function normally with the exception of the `<AUTOEXEC>` features that require a persistent repository data set.

- **Data type:** Not applicable.
- **Parent element:** `<OPTIONS>`
- **Child elements:** `<NAME>`, `<STORCLAS>`, `<VOLUMES>`, `<MGMTCLAS>`, `<DATACLAS>`, and `<IGNORE>`

---

**NAME**

*(optional)* The `<NAME>` element specifies the data set name for the DPR repository VSAM cluster. This value must specify a valid data set name:

- If the value specifies the name of an existing DPR repository VSAM cluster, DPR repository services use that cluster for the repository.

- If the specified data set does not exist, DPR repository services define the new cluster automatically by using the `<STORCLAS>` or `<VOLUMES>` options to determine physical allocation attributes.
If you do not provide a value in the <NAME> element, DBC issues a warning message during initialization, and the repository services are unavailable.

Only one active DBC subsystem can use a given DPR repository VSAM cluster.

You can share a single DBCPARMS parameters file for multiple DBC instances. To do so, specify DBC system variables within the <NAME> element to ensure that each DBC subsystem uses a unique repository data set name. DBC resolves the variables to their symbolic values. To enable variable substitution, prefix the system variables with an ampersand (&) and suffix them with a period (.)

**Example**

Assume that you use the system variables in a <NAME> element as shown in the following statement:

```
<NAME>BMC.&SSID..&SMFID..&MVSNAME..REPOS</NAME>
```

Also, assume that the symbolic values of SSID, SMFID, and MVSNAME are **DBC1**, **SYSP**, and **MVSPROD**, respectively.

The variable substitution resolves to the following data set name:

```
BMC.DBC1.SYSP.MVSPROD.REPOS
```

**Data type:** VARCHAR(44)

**Parent element:** <DPRREPOS>

**Child elements:** None.

### STORCLAS

*(optional)* The <STORCLAS> element specifies the SMS storage class to be used for the DPR repository VSAM cluster when the DPR defines the cluster. The DPR defines the cluster automatically if the <NAME> element specifies the name of a cluster that has not yet been defined. Specify this element only if SMS is active and the cluster is to be SMS-managed. For more information, see the STORAGECLASS parameter of the DEFINE CLUSTER command in the IBM DFSMS Access Method Services documentation.

**Data type:** VARCHAR(8)

**Parent element:** <DPRREPOS>

**Child elements:** None.

### VOLUMES

*(optional)* The <VOLUMES> element specifies the volumes on which the DPR repository VSAM cluster is to be defined when the DPR defines the cluster. The DPR defines the cluster automatically if the <NAME> element specifies the name of a cluster that has not yet been defined.
This element is required only if you do not specify if a value for <STORCLAS>.

The <VOLUMES> value can specify up to 59 volumes, where each volume is delimited by a comma (,). The value can specify a volume serial number or, if the cluster is to be SMS-managed, an asterisk (*), which lets SMS choose the volume. For more information, see the VOLUMES parameter of the DEFINE CLUSTER command in the IBM DFSMS Access Method Services documentation.

**Data type:** VARCHAR(412)

**Parent element:** <DPRREPOS>

**Child elements:** None.

### MGMTCLAS

*Optional* The <MGMTCLAS> element specifies the management class for the repository data set to be created.

The <MGMTCLAS> element is used only if the repository data set does not yet exist.

**Data type:** VARCHAR(8)

**Parent element:** <DPRREPOS>

**Child elements:** None.

### DATAACLAS

*Optional* The <DATAACLAS> element specifies the data class for the repository data set to be created.

The <DATAACLAS> element is used only if the repository data set does not yet exist.

**Data type:** VARCHAR(8)

**Parent element:** <DPRREPOS>

**Child elements:** None.

### IGNORE

*Optional* The <IGNORE> element tells DBC to ignore the repository entries for the identified product at startup. Doing so allows the rest of the products running within the DBC subsystem to start normally without starting the specified product. You can include multiple <IGNORE> elements.

when DBC is started

**Data type:** Not applicable.

**Parent element:** <DPRREPOS>

**Child elements:** <PRODUCT>, <FMID>, and <PIID>
PRODUCT

(required) The <PRODUCT> element identifies a unique BMC product code.

Data type: CHAR(3)
Parent element: <IGNORE>
Child elements: None.

FMID

(required) The <FMID> element identifies the FMID of the product code release.

Data type: CHAR(7)
Parent element: <IGNORE>
Child elements: None.

PIID

(optional) The <PIID> element is optional and identifies a unique product instance within the context of one product code and FMID.

Data type: VARCHAR(16)
Parent element: <IGNORE>
Child elements: None.

Stopping the DBC subsystem

Use the following procedure to stop the DBC subsystem through an IBM z/OS console command. You identify the DBC subsystem to stop through the subsystem ID (ssid) or XCF group value (group).

WARNING

Shutting down the DBC subsystem also shuts down all associated Subzero servers used by BMC Subsystem Optimizer for zEnterprise. If a server shuts down while active connections exist between the client and its remote subsystem, abends can occur in the Subzero, client, and subsystem address spaces.

Before you terminate the DBC subsystem or any Subzero address space, execute the processes and procedures that properly terminate a connection between a client and its connected subsystem.

To stop the DBC subsystem

1. Enter one of the following z/OS commands:
   - ssid STOP
   - group STOP
   - MODIFY ssid,STOP
Example

Assume that a DBC subsystem has an SSID of DBC1. You can stop this subsystem by issuing the STOP command as a MODIFY command:

\[ \text{F DBC1,STOP} \]

WARNING

Some product processes might require an extended amount of time to terminate successfully. If DBC shutdown is delayed, review DBCPRINT for the following message to determine the cause of the delay:

\[ \text{BMCDBC0232I… DBC is waiting for product product\(\text{Name}\) to shutdown.} \]

If the DBC does not shut down after a reasonable amount of time, you can use the following command to terminate the delayed process:

\[ \text{ssid STOP,FORCE} \]

Use caution when doing so; terminating a process that is still in use can have unpredictable results. BMC recommends that you use the STOP,FORCE command only if preceding attempts to stop the DBC have failed.

DBC commands

You can interact with the DBC subsystem by using any of the following commands.

The \textit{ssid} variable refers to the subsystem ID of the DBC subsystem.

Table 5: DBC commands

<table>
<thead>
<tr>
<th>Commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/\textit{ssid} DBC,\texttt{DIAG}</td>
<td>Sends diagnostic information to the DBCPRINT and DBCTRACE data sets BMC Support might direct you to execute this command to assist in diagnosing problems.</td>
</tr>
</tbody>
</table>
| /\textit{ssid} DBC,\texttt{DB2C,ssid,command} | Executes a DB2 command in the DB2 subsystem identified by "ssid" and returns the result to DBCPRINT. This command can be used to execute any native DB2 command that the user is authorized for.  
\textbf{Note:} This command cannot be used for DSN subcommands. |
| /\textit{ssid} DBC,\texttt{RESETPRINT} | Closes the DBCPRINT data set and allocates a new one in order to clear data from DBCPRINT |
| /\textit{ssid} DBC,\texttt{RESETTRACE} | Clears the DBCTRACE data set and allocates a new one |
| /\textit{ssid} STOP | Stops the DBC subsystem |
| /\textit{ssid} DBC,\texttt{TRACEOFF} | Disables DBC tracing |
### Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| `/ssid DBC,TRACEON` | Enables DBC tracing  
DBC writes trace data to the DBCTRACE data set. The DBCTRACE data set is dynamically allocated.  
**WARNING:** If you leave tracing enabled, be aware that the DBCTRACE data set can grow to a significant size. BMC recommends using caution when using the TRACEON command. |
| `/ssid DBC,TRACEFLUSH` | Forces unwritten trace information to the DBCTRACE data set  
This command is useful if trace was disabled. DBC writes any trace data that exists in an internal buffer that is used to maintain trace information. |
| `/ssid DBC,PRODCMDS,ppp` | Displays all commands for product `ppp` (*Displays only commands that conform to the ppp,command format*). |
| `/ssid DBC,DISPLAY,object` | The details of the DBC, DISPLAY command for each object type are documented in the following table. |

### Table 6: DISPLAY Command Options

<table>
<thead>
<tr>
<th>DISPLAY object (alternative syntax)</th>
<th>Description</th>
</tr>
</thead>
</table>
| PRODUCTS (PRODUCTS, PRD, PDXD) [,ppp] | Displays the Product Definition XML Document (PDXD) from the DBC repository. If no parameters are specified, information is displayed for all products  
If the three-character product code is specified, DBC displays information for only that product. If multiple PIIDS exist for that product code, they are all displayed. |
| CMDS [,ppp] | Displays all operator command definitions. If no parameters are specified, commands for all products are displayed.  
If the three-character product code is specified, DBC displays only commands for that product. |
| EVENT (EV),eventName [subid] | Displays all subscription information the named event  
If the `subid` is specified, only information for the named subscription is displayed. |
<p>| CMD,cmdtext | Displays the ACTION XML for the named command. |</p>
<table>
<thead>
<tr>
<th>DISPLAY object (alternative syntax)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGENTS (AG) [,ppp]</td>
<td>Displays all agents that are currently executing. This command generates message BMCDBC0126I, which indicates the TCB address of the agent and whether it is a job step TCB. If no parameters are specified, this command displays information for all products. If the three-character product code is specified, displays currently execution agents for only the product specified.</td>
</tr>
<tr>
<td>PROCESSES</td>
<td>Displays all active PROCESSES. If no other parameters are specified, PROCESSES for all products are displayed. If the three-character product code is specified, displays currently active PROCESSES for only the product specified.</td>
</tr>
<tr>
<td>DB2 [,ssid][,LOCAL</td>
<td>ALL ]</td>
</tr>
<tr>
<td>IMS [,imsid][,LOCAL</td>
<td>ALL ]</td>
</tr>
<tr>
<td>DISPLAY object (alternative syntax)</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>MQ [,ssid][,LOCAL</td>
<td>ALL ]</td>
</tr>
<tr>
<td>SRBS [,ppp ]</td>
<td>Displays currently active SRBs. If no parameters are specified, SRBs for all products are displayed. If the three-character product code is specified, displays currently active SRBs only for the product specified.</td>
</tr>
<tr>
<td>STATS [,TCBS,name] [,MEM][,QUEUES]</td>
<td>If no parameters are specified, this command displays all available statistics. name displays statistics for only the named task. MEM displays memory usage statistics. QUEUES displays queue usage statistics.</td>
</tr>
<tr>
<td>DSNS</td>
<td>Displays all data sets currently allocated and specifies if the data set allocation results from STEPLIB, product, or agent definition.</td>
</tr>
<tr>
<td>SOCKETS</td>
<td>Displays the state of all sockets currently in use.</td>
</tr>
</tbody>
</table>

# DBC security

This section introduces security for DBC.

Consider the following requirements and features of DBC security:

- **DBC must meet the following UNIX requirements:**
  
  — Write and execute access to the /tmp directory.
  
  — Update access to the FSACCESS (UNIX file system access check) resource class.
DBC must be authorized to create an Extended MCS Console.

DBC uses the standard System Authorization Facility (SAF) interface to communicate with an External Security Manager (ESM). DBC is compatible with ESMs that support the SAF interface (including CA Technologies CA-ACF2 and CA-Top Secret products).

The DBC security interface is compatible with the IBM Resource Access Control Facility (RACF) Version 1.9 or later. DBC issues security calls directly to the SAF interface.

For more information, see “Configuring RACF security for the DBC subsystem” on page 38.

DBC does not require you to define resources (that is, internal control points) to an ESM. However, if the security parameter <ALLOW_SAF_RC4> is set to NO and no resource names have been defined, all DBC requests fail with an authorization error. If <ALLOW_SAF_RC4> is set to NO, you must define DBC resource names so that access can be granted or denied as appropriate.

DBC security control points include commands that are issued from an IBM z/OS system console and from the IBM System Display and Search Facility (SDSF). The DBC subsystem command processor extracts the user ID from the ACEE that is associated with the console address space and propagates this value through the system. Appropriate ESM customization is required to allow operator authorization to DBC commands.

DBC stores its security parameters in the DBCSECUR data set.

DBC security parameters are independent of all other DBC parameters. This physical separation allows the security administrator to implement independent and discrete access to the security parameters (DBCSECUR) and potentially more general access to the subsystem parameters (DBCPARMS).

For more information about security parameters, see “DBC security parameters” on page 41.

Controlling access to DBC resources

The DBC subsystem always issues a Security Access Facility (SAF) security call for internal resource names. By default, if those resource names are not defined to an ESM, access to undefined DBC resources (internal functional control points) is granted for all users who can communicate to the DBC service address space. This behavior occurs because the value of the DBC <ALLOW_SAF_RC4> security parameter defaults to YES.

If you want to restrict access to DBC services that have not been defined as a resource to SAF, you must set the value of the <ALLOW_SAF_RC4> security parameter to NO. Doing so denies access to all undefined resources.
Figure 3 on page 38 shows an example of the ALLOW_SAF_RC4 value.

Figure 3: Example of the ALLOW_SAF_RC4 value

```xml
<DBCSECUR>
  <RESOURCE_NAME>
    ...
  </RESOURCE_NAME>
  <ALLOW_SAF_RC4>NO</ALLOW_SAF_RC4>
</DBCSECUR>
```

You control this value through DBC security parameters defined in a file identified through the DBCSECUR DD statement in the JCL procedure for the DBC started task.

- To specify the location of the DBC security parameters, see “DBC startup parameters” on page 23.
- To change the value of the `<ALLOW_SAF_RC4>` security parameter, see “DBC security parameters” on page 41.

## Configuring RACF security for the DBC subsystem

The security administrator for a site should perform the following tasks to authorize the DBC subsystem:

- “Authorizing the DBC started task procedure” on page 38
- “Authorizing a user with access to the DBC services” on page 39
- “Creating a SAF resource class (optional)” on page 39
- “Defining a DBC control point resource profile” on page 40

### Authorizing the DBC started task procedure

The DBC subsystem acquires security authorization from the USERID associated with the started task procedure that starts the DBC subsystem. Users must authorize the DBC started task.

**To authorize the DBC started task**

1. Associate the DBC started task with a USERID that has the appropriate security access.

   The DBC address space requires access to z/OS UNIX System Services (USS). Consequently, you must associate an OMVS segment with the USERID for the DBC subsystem.
The following example uses a RACF command to define a GROUP of DBCGRP:

```
ADDUSER DBCGRP OMVS(GID(groupIdentifier))
```

The following example uses a RACF command to define a USERID of DBCUSR:

```
ADDUSER DBCUSR DFLTGRP(SYSMGMT) OWNER(SYSPROG)
```

2 Associate the USERID with the procedure name for the DBC started task.

```
Example
```

In the following example, the RACF command uses procedure name DBCSTC:

```
RDEFINE STARTED DBCSTC.* STDATA(USER(DBCUSR) GROUP(DBCGRP))
```

3 To ensure that RACF recognizes these security updates, issue the following command:

```
SETROPTS RACLIST(STARTED) REFRESH
```

**Authorizing a user with access to the DBC services**

To access DBC services, batch and TSO users require access to z/OS UNIX System Services (USS) and must be associated with an OMVS segment.

```
Example
```

The following example uses a RACF command to modify the appropriate group for a user:

```
ALTGROUP groupName OMVS(GID(groupIdentifier))
```

The following example uses a RACF command to modify the user’s profile:

```
ALTUSER userName OMVS(UID(userIdentifier))
```

**Creating a SAF resource class (optional)**

This section explains how to create a new RACF resource class. Performing this task is not necessary if the predefined RACF FACILITY class is appropriate for all DBC resource definitions.

```
Note
```

This section provides general instructions. Your site might have additional considerations. BMC recommends consulting your security administrator when creating your SAF resource class.

**To update the RACF resource class descriptor table (ICHRRCDE)**

1 Code the ICHERCDE macro for each required resource class.

```
TITLE 'RACF RESOURCE CLASS DESCRIPTOR TABLE - ICHRRCDE'
```

```
class ICHERCDE CLASS=class.
```

2 Assemble the source and link edit the resulting object module to replace the current ICHRRCDE load module.

**To update the RACF router table (ICHFR01)**

1 Code the ICHRFRTB macro for each required resource class.

```assembler
* TITLE 'RACF ROUTER TABLE - ICHRFR01'
* class ICHRFRTB CLASS=class.
  ACTION=RACF
* ICHRFRTB TYPE=END
* END
```

2 Assemble the source and link edit the resulting object module to replace the current ICHRFR01 load module.

3 If necessary to activate the RACF table changes, perform an IPL of the system.

4 Issue the following RACF command for each new resource class:

```
SETROPTS CLASSACT(class)
```

5 *(optional)* If you use generic profiles, issue the following RACF commands for each new resource class:

```
SETROPTS GENERIC(class)
SETROPTS GENCMD(class)
```

**Defining a DBC control point resource profile**

The DBC subsystem and its components automatically call the SAF router to check user authorization to various services.

These services are identified by internal functional control points and are externally associated with a resource name. You control user access to the DBC component services by granting or denying authorization to the resource names that are associated with these internal functional control points. To control access to these services, you must define these resource names to the ESM.
To define a DBC control point resource profile

1 Define the resource profile (that is, the resource name) to the RACF ESM by using one or more RDEFINE FACILITY commands.

Example

The following example protects access to various resource categories for a DBC subsystem. The profile applies to LPARs named PROD. You set the PROD value through the < CONTEXT> XML element in the DBC security parameters. The profile also applies to product codes DBC and DPR (inherent components of the DBC subsystem) and ABC (which relates to a DPR-initialized product with the 3-byte product code ABC). For example, the BMC System and SQL Performance products for DB2 use DBC, DPR, LGC, and NGL.

RDEFINE FACILITY (BMC.DBC.PROD.*) UACC(NONE)
RDEFINE FACILITY (BMC.DPR.PROD.*) UACC(NONE)
RDEFINE FACILITY (BMC.ABC.PROD.*) UACC(NONE)

The next example defines a generic profile that protects all currently defined subsystem resources and future resources that are associated with products that you have not yet defined to the DPR component of DBC:

RDEFINE FACILITY (BMC.*.PROD.*) UACC(NONE)

2 Activate the resource class by issuing one of the following commands:

- SETROPTS CLASSACT(FACILITY)
- SETROPTS CLASSACT(FACLITY) RACLIST(FACILITY) (to maintain profiles in memory)

3 (optional) Enable generic profile checking for the FACILITY class:

SETROPTS GENERIC(FACILITY)

DBC security parameters

The DBCSECUR DD statement, which is specified in the startup JCL for the DBC subsystem, identifies the security parameters data set for the DBC subsystem. The security parameters are optional. Each security parameter has a default value that applies if you do not specify the DBCSECUR DD statement, or if you omit a particular security option from the parameters file. Thus, you are not required to specify these parameters to use the DBC security features.

You do not specify the security parameters with the main DBC startup parameters identified in the DBCPARMS DD statement. You can choose to administer and secure the security parameters separately from the main DBC startup parameters by implementing RACF data set name security. For more information about the started task, see “Started task for the DBC subsystem” on page 21.
Structure of the XML stream

Figure 4 on page 42 shows a sample security parameters file.

**Figure 4: Structure of the DBCSECUR XML stream**

```
<DBCSECUR>
  <RESOURCE_NAME>
    <HLQ>BMC</HLQ>
    <CONTEXTS>
      <CONTEXT>
        <SMFID>MVSA</SMFID>
        <TO_VALUE>PROD</TO_VALUE>
      </CONTEXT>
      <CONTEXT>
        <SMFID>MVSB</SMFID>
        <TO_VALUE>TEST</TO_VALUE>
      </CONTEXT>
    </CONTEXTS>
  </RESOURCE_NAME>
  <RESOURCE_CLASS>
    <COMPONENT>DBC</COMPONENT>
    <COMMAND>MYCLASS</COMMAND>
  </RESOURCE_CLASS>
  <RESOURCE_CLASS>
    <COMPONENT>DPR</COMPONENT>
    <COMMAND>MYCLASS</COMMAND>
  </RESOURCE_CLASS>
  <SUBSYS>DBCS</SUBSYS>
  <ALLOW_SAF_RC4>NO</ALLOW_SAF_RC4>
</DBCSECUR>
```

**DBCSECUR elements**

*(optional)* The `<DBCSECUR>` element is the root-level element of the DBCSECUR structure.

**Data type:** Not applicable.

**Child elements:** `<RESOURCE_NAME>`, `<RESOURCE_CLASS>`, `<SUBSYS>`, `<ALLOW_SAF_RC4>`, and `<DB2AUTH>`

**RESOURCE_NAME**

*(optional)* The `<RESOURCE_NAME>` element contains the options for the customizable resource name nodes.

**Data type:** Not applicable.

**Parent element:** `<DBCSECUR>`

**Child elements:** `<HLQ>` and `<CONTEXTS>`

**HLQ**

*(optional)* The `<HLQ>` element specifies a value for the HLQ node of the resource name structure. This value defaults to BMC.

**Data type:** VARCHAR(8)

**Parent element:** `<RESOURCE_NAME>`

**Child elements:** None.
CONTEXTS

*(optional)* The `<CONTEXTS>` element contains one or more context specifications.

**Data type:** Not applicable.
**Parent element:** `<RESOURCE_NAME>`
**Child elements:** `<CONTEXT>`

CONTEXT

*(required)* The `<CONTEXT>` element specifies a value for the context node of the resource name structure. This option defaults to the SMFID that is associated with the IBM z/OS image on which the DBC subsystem is active. The `<CONTEXT>` element must specify a `<SMFID>` and `<TO_VALUE>` element.

**Data type:** Not applicable.
**Parent element:** `<RESOURCE_NAME>`
**Child elements:** `<SMFID>` and `<TO_VALUE>`

SMFID

*(required)* The `<SMFID>` element specifies the SMFID that is associated with the z/OS image on which the DBC subsystem is active.

**Data type:** VARCHAR(4)
**Parent element:** `<CONTEXT>`
**Child elements:** None.

TO_VALUE

*(required)* The `<TO_VALUE>` element specifies the value that the DBC subsystem uses for the context node of the resource name.

**Data type:** VARCHAR(4)
**Parent element:** `<CONTEXT>`
**Child elements:** None.
**Example**

The following example tells the DBC subsystem to use PROD as the context node of the resource name if the subsystem is started on the z/OS image with an SMFID of LPR1:

```xml
<CONTEXT>
  <SMFID>LPR1</SMFID>
  <TO_VALUE>PROD</TO_VALUE>
</CONTEXT>
```

If not started on this z/OS image, the DBC subsystem ignores the `<CONTEXT>` specification. This behavior allows you to define a single SAF options file for multiple DBC subsystems that run on different z/OS images.

By modifying the `<CONTEXT>` element to point a specific LPAR to a logical name, you can define RACF resource names that are consistent across multiple LPARs.

---

**RESOURCE_CLASS**

*(optional)* The `<RESOURCE_CLASS>` element allows you to customize the SAF resource class that is associated with internal DBC security control points. If omitted, the RACF resource class for all DBC commands (and associated components) defaults to the FACILITY class.

---

**Note**

This value does not affect the SAF resource class for DPR-initialized product objects. You can customize those classes by using the `<SAFCLASS>` XML tag in the product definition XML document.

---

In the sample shown in Figure 4 on page 42, all DBC and DPR command resource profiles must be defined in RACF resource class `MYCLASS`.

The `<COMPONENT>` and `<COMMAND>` subelements are required only if you use the `<RESOURCE_CLASS>` element.

**Data type:** Not applicable.

**Parent element:** `<DBCSECUR>`

**Child elements:** `<COMPONENT>`

---

**COMPONENT**

*(required)* The `<COMPONENT>` element identifies the specific component for which you are defining the resource class.

**Data type:** CHAR(3)

**Parent element:** `<RESOURCE_CLASS>`

**Child elements:** `<COMMAND>`
COMMAND

*(required)* The `<COMMAND>` element identifies the SAF resource class name that is used in all SAF security calls for commands issued from the associated component.

- **Data type:** VARCHAR(8)
- **Parent element:** `<COMPONENT>`
- **Child elements:** None.

SUBSYS

*(optional)* The `<SUBSYS>` element specifies the value to be passed to SAF on each authorization check to the SUBSYS parameter on the RACROUTE macro. The value defaults to DBCS.

- **Data type:** VARCHAR(8)
- **Parent element:** `<DBCSECUR>`
- **Child elements:** None.

---

**Note**

DBC uses BMCDBC as the application name that is passed to SAF through the APPL parameter on the RACROUTE REQUEST=AUTH macro call. This parameter specifies the name of the application that is making the authorization request. The RACROUTE service makes the parameter available to the installation exit routine, or any routines that the service invokes.

ALLOW_SAF_RC4

*(optional)* The `<ALLOW_SAF_RC4>` element specifies whether the DBC subsystem allows access to a given resource if SAF returns return code 4. SAF returns 4 if a security decision could not be made.

Valid values are **YES** and **NO**:

- **YES** *(the default)* tells the DBC subsystem to allow user access to a resource if SAF returns return code 4.
- **NO** tells the DBC subsystem not to allow user access to the resource.

- **Data type:** VARCHAR(3)
- **Parent element:** `<DBCSECUR>`
- **Child elements:** None.

---

**Note**

The IBM RACROUTE macro reference documentation (SA22-7692-04) documents the SAF return codes.

---
The DB2 Product Configuration (LGC) component separates product or solution installation from configuration. This chapter describes how to use the LGC interface, how to create and manage option sets, and how to issue commands to the LGC component.

LGC utilizes a client/server model. The DBC is the server, and the ISPF online interface and LGCUTIL batch utility are the clients.

Through its online interface, LGC simplifies configuration as follows:

- Saves time by:
  - Providing default option values for you, but allowing you to change them if needed
  - Consolidating the process into scrollable panels that feature a simple navigation model (expanding sections on the panel)
  - Retaining data for subsequent installations and upgrades

- Validates any option values that you enter

- Lets you link to the configuration interface from within a product or solution

- Captures option value changes that you make after installation and initial configuration

- Provides maintenance via PTFs

- Supports multiple screen sizes (for example, MOD 2, MOD 4, MOD 5, or custom-sized terminals)

LGC creates XML files called option sets to contain all of your product’s configuration values. Some products allow multiple option sets. For those products, you can use different option sets, or configurations, to run different jobs.
Overview of the LGC interface

DB2 Product Configuration (LGC) provides ISPF panels in which you can enter or change product or solution option set values.

The Product Option Sets panel (Figure 5 on page 48) is one of the basic panels that you use to create new option sets. It lists each product and solution for which you can configure option sets.

Figure 5: Product Option Sets panel

Expandable sections

The LGC interface provides expandable and collapsible sections that group options into categories. Expandable sections are identified with a plus sign (+) to the left of the section’s title.

Navigation in sections

Figure 6 on page 48 shows an example of the Product Option Sets panel, which has a section for each product.

Figure 6: Product Option Sets panel (not expanded)
In this panel, you can expand a section by placing your cursor on the adjacent plus sign (+) and pressing Enter. For example, expanding a section displays your option sets for that product or solution (Figure 7 on page 49).

**Figure 7: Product Option Sets panel (expanded to show option sets)**

<table>
<thead>
<tr>
<th>Product Option Sets</th>
<th>Command ==&gt;</th>
<th>Scroll ==&gt;</th>
<th>CSR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solution/Product</td>
<td>Version Changed</td>
<td>More:</td>
</tr>
<tr>
<td></td>
<td>- BMC Workbench</td>
<td>V01.0.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>_ GUDOPT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- PACLOG for DB2</td>
<td>V11.1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>_ ALM$OPTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- RECOVERY MANAGER for DB2</td>
<td>V11.1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>_ ARM$OPTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- System and SQL Performance for DB2</td>
<td>V11.1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>_ DCPLEX</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To collapse an expanded section, place your cursor on the minus sign (-) and press Enter. To expand or collapse multiple sections, type S on all of their signs and press Enter. Attempting to expand an empty section simply changes the + sign to a - sign.

**Instance indicators for a section**

If a section title has a number after it in parentheses, more than one instance for that section exists.

For example, Figure 8 on page 49 shows an LGC interface panel for the SQL Performance solution.

**Figure 8: Example of instance indicators**

<table>
<thead>
<tr>
<th>File</th>
<th>Filter</th>
<th>Confirm</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command ==&gt;</td>
<td>DBCRIOPT - DOMPLEX FOR DB2R101</td>
<td>Scroll ==&gt;</td>
<td>CSR</td>
</tr>
<tr>
<td>Filter: Modified</td>
<td>More:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ DOMPLEX Parameters</td>
<td>Parameters that apply to entire DOMPLEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ Data Collector List (1)</td>
<td>Data Collector(DBC) subsystems in DOMPLEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ DB2 Monitor List (12)</td>
<td>DB2 Sub-systems to be monitored</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ OutGp DCID DspSize (6)</td>
<td>Output Groups - valid range: 001-256</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this example, the option set contains:

- One instance of the Data Collector List section
Twelve instances of the DB2 Monitor List section

Six instances of the OutGp DCID DspSize section

If no number exists, that section is unique and can have only one set of values.

If you expand a section that has an instance indicator, each instance is displayed. For example, Figure 9 on page 50 shows six output groups that are associated with the OutGp DCID DspSize section. This example also shows the first output group expanded to show its subsections.

Figure 9: Example of multiple instances within a section

File  Filter  Confirm  Help
-------------------------------------------------------------------------------
DBCRIOPT - DOMPLEX FOR DBCR1101
Command ===>                                                  Scroll ===> CSR
Filter: Off                                          More:
+ DOMPLEX Parameters               Parameters that apply to entire DOMPLEX
+ Data Collector List (1)          Data Collector(DBC) subsystems in DOMPLEX
+ DB2 Monitor List (12)            DB2 Sub-systems to be monitored
- OutGp DCID DspSize (6)           Output Groups - valid range: 001-256
  + Data Classes                   Specify IFCIDs to be stored in this group
  + NGL LOGSET Parameters          LOGSET attributes used by this group
    DB2 Subsystem IDs...: *
  - 012 N11J 20                     Dataspace buffer size: 1-2000 (MB)
  + 014 N11J 20                     Dataspace buffer size: 1-2000 (MB)
  + 015 N11J 20                     Dataspace buffer size: 1-2000 (MB)
*******************************************************************************
End of List

Note

If a field has a greater than sign (>) to the left of it, you can zoom the field to see more details by positioning the cursor on > and pressing Enter.

For example, assume that the panel includes the following field entry:

> DB2 IFCIDs to be discarded

You can position the cursor on > and press Enter to see the entire list of discarded IFCIDs.

Drop-down menus

The top of an LGC interface panel provides drop-down menus (File, Filter, Confirm, and Help). You can use these menus to manage your files, filter the option sets that are displayed in expanded sections of the panel, and obtain Help.
Action list

The action list at the bottom of an LGC interface panel indicates allowable actions or line commands for that panel. From an option sets panel, you can perform the following actions on option sets:

- Browse (B)
- Edit (E)
- Copy (C)
- Delete (D)
- Insert (I)
- Rename (R)

To perform an action, type the corresponding letter next to the option set upon which you want to perform that action. For example, to edit an option set, enter E next to the option set’s name.

Online Help

LGC provides online Help for panels, fields in panels, and messages.

You can access Help in the following ways:

- To access Help for a panel, use the Help drop-down menu and then select General Help or Product Help. The Product Help choice will not be available unless you are editing or browsing an option set.

- To access Help for an entry field, place the cursor on the field and press F1.

- To access additional information about any message that the LGC interface displays, press F1 when the message is displayed. The interface will then display a longer form of the message.

Working with option sets

LGC creates XML files called option sets to contain all of your product’s configuration values. Some products allow multiple option sets. For those products, you can use different option sets, or configurations, to run different jobs.
Option set names can be up to eight characters long and can contain any of the following characters:

- Uppercase A through Z
- 0 through 9
- Special characters !, @, $, or #

The first character of the name must be an uppercase alphabetic character (A through Z) or one of the following symbols: @, $, or #. The name must conform to PDS member naming standards.

Note
Individual BMC products that use LGC might include different naming standards. However, the information in the noted guidelines describes the default standard.

For more specific information about the option sets that your product uses and the values that the option sets contain, see the documentation for your product.

This topic explains how to:

- Access the online LGC interface
- Create a new option set
- Edit an existing option set
- Rename an option set
- Delete an obsolete option set
- Filter the values to display within an option set

Accessing the online interface

Use this procedure to access the online interface for managing option sets.

1. Invoke the appropriate CLIST, as follows:

   - (for System or SQL Performance products) Use DOMCLIST or SPDCLIST (generated in the installation JCL data set and copied to the UBMCCCLIB data set) or hyperlink from MainView to launch the product.
(for all other products) Use the LGCISPF CLIST (generated in the installation JCL data set and copied to the product UBMCCLIB or UBMCSAMP data set).

2 If you invoked DOMCLIST or SPDCLIST from your UBMCCLIB data set (or hyperlinked from MainView), complete this step:
   a Select Administration => DOMPLEX Option Sets.
   b Add the DOMPLEX option set name:
      ▪ If you are migrating from an earlier release, select the DOMPLEX option set name that matches the DOMPLEX name used previously.
      ▪ For a new installation, select the option set that was created as part of the installation job. This name will match the DOMPLEX parameter in the $495SDOM job.

   **Note**
   By default, the product CLISTs set FILTOFF and show all values.

3 If you invoked the LGCISPF CLIST, complete this step:
   a From the DB2 Product Configuration Main Menu, select Manage Product Options.
      The Product Option Sets panel shows all products that use option sets.
   b Select the product to expand the list of option sets.
   c Select the specific option set that you want to edit.

   **Note**
   By default, the LGCISPF panel shows only the modified options that are different from the shipped defaults. To see all options when editing the option set, type FILTOFF on the COMMAND line.

Creating a new option set

This task provides a high-level overview of creating a new option set. For more specific instructions for a particular product, refer to the documentation for your product.
To create an option set

1 On the Product Option Sets panel, create a new option set by using one of the following methods:
   - Enter I next to the product or solution name for which you want to create an option set.
   - Enter C next to an option set that you want to copy and modify to create a new option set.

2 Press Enter.

3 Expand each section and review or change the values.
   
   Tip
   For repeatable sections, you can:
   - Enter I next to the section to insert a new section that uses all default values.
   - Enter R next to a section to duplicate the section with its current values. You can then edit the repeated section to create a unique section.

   Repeatable sections include a number in parentheses behind the section name.

4 Save your option set by pressing F3.
   
   Note
   You can stop an editing session before you are finished and save your changes, even if the option set still has validation errors. To do so, press F3 and select Save With Errors.
   Do not try to use an option set that contains validation errors. Doing so can cause unpredictable results. Always ensure that your option set is valid before using it.

Editing an option set

You can edit the option sets by executing the product’s CLIST or the LGCISPF CLIST. All methods invoke the same LGC dialog.

Before you begin

The DBC started task must be running and the LGC agents must be active.
To edit an option set

1. On the Options Set panel, type E next to the option set that you want to edit and press Enter.

2. Expand each section and review or change the values.

   **Tip**
   For repeatable sections, you can:

   - Enter I next to the section to insert a new section that uses all default values.
   - Enter R next to a section to duplicate the section with its current values. You can then edit the repeated section to create a unique section.

   Repeatable sections include a number in parentheses behind the section name.

   **Note**
   While editing the option set, you might receive validation error messages if fields contain invalid entries. Validation errors change the filter setting to Filter: Invalid, and only fields that have validation errors are displayed. When you correct an validation error, the associated field is no longer displayed on the panel. When you have corrected all the validation errors, the filter reverts to its previous setting (for example, Filter: Required or Filter: Off).

   You can manually change the filters if you want to see all options. For more information about filtering, see “Filtering option sets” on page 58.

3. Save your option set by pressing F3.

Example of working with default values for option sets

This example explains how to create default values for a section in an option set. Doing so allows you to define your own default values so that whenever you insert new instances of a section, you will have fewer values to update.

The values that you specify for section defaults are used whenever you insert a new instance. If you change a value in a section default, any value in an instance that matched that value prior to the change is eligible for being updated to the new value. This process allows you to change the values in multiple instances of a section at once, without opening and closing each instance.
1 Access the LGC panel for editing the option set (as described in Accessing the online interface on page 52).

Figure 10 on page 56 shows an example of a panel for a new DOMPLEX option set.

**Figure 10: Panel for a new option set**

<table>
<thead>
<tr>
<th>File</th>
<th>Filter</th>
<th>Confirm</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command ===&gt;</td>
<td>DBCRIOPT - DOMPLEX FOR DBCRI101</td>
<td>Scroll ===&gt; CSR</td>
<td></td>
</tr>
</tbody>
</table>

Filter: Modified
+ DOMPLEX Parameters
+ Data Collector List (3)
+ DB2 Monitor List (16)
+ OutGp DCID DspSize (9)

+/- expandable section (enter ? for action menu), > zoomable field

**Tip**
The number in parentheses behind the section name indicates the number of instances that exist for a section. Any section that has an instance count of 0 or more is eligible for using section default values. (In the example panel, the DOMPLEX Parameters section does not contain any instances.)

2 To update section default values, type E next to the relevant section.

The LGC interface displays a panel that allows you to define the default values pertinent to that section.

For example, typing E next to **DB2 Monitor List** displays the panel shown in Figure 11 on page 56.

**Figure 11: Example panel for defining defaults**

<table>
<thead>
<tr>
<th>File</th>
<th>Filter</th>
<th>Confirm</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command ===&gt;</td>
<td>DB2 Monitor List - Section Defaults</td>
<td>Scroll ===&gt; CSR</td>
<td></td>
</tr>
</tbody>
</table>

Filter: Off
DB2 Subsystem ID . . . . . . . . . . . . . . . SSID
Is this a production DB2? . . . . . . . N (Y=Yes,N=No)
Monitor with MAINVIEW for DB2 . . . . . Y (Y=Yes,N=No)
Monitor with Pool Advisor/System Perf. . Y (Y=Yes,N=No)
Monitor with APPTUNE . . . . . . . . . . . . Y (Y=Yes,N=No)
Dynamic Explain plan name . . . . . . . . DAA1111D1
> DB2 IFCIDs to be traced automatically
> DB2 IFCIDs to be discarded
> BMC IFCIDs to be discarded
Class 2-In-DB2 elapsed timing info . . . Y (Y=Yes,N=No)
Class 3-DB2 suspend timing info . . . Y (Y=Yes,N=No)
Class 5-Time spent doing IFI requests Y (Y=Yes,N=No)
Class 7-DB2 events (packages, DBRMs). Y (Y=Yes,N=No)
Class 8-Wait time for packages . . . . Y (Y=Yes,N=No)
3 Make the changes to the option set.

For example, assume that all of the Class fields were changed to N and the previous value of Y matched the existing instances of the defined DB2 subsystems.

4 Press F3 to save the changes and display the Figure 12 on page 57 panel.

**Figure 12: Apply Sections Defaults Updates**

```
LGCW1004                                                Changes applied
Updates to the model can be applied to the following 12 rows.
_ Enter "/" to select all rows or select individual rows in the list.
DB2 Monitor List                                           More:   +
_ DB2A           DB2 SSID
_ DB2B           DB2 SSID
_ DB2C           DB2 SSID
_ DB2D           DB2 SSID
_ DB2E           DB2 SSID
_ DB2F           DB2 SSID
_ DB2G           DB2 SSID
_ DB2H           DB2 SSID
_ DB2I           DB2 SSID
_ DB2J           DB2 SSID
_ DB2K           DB2 SSID
_ DB2L           DB2 SSID
```

Press END to update the selected rows.

5 Take one of the following actions:

- To change the value from Y to N for all instances, type / next to Enter "/" to select all rows or select individual rows in the list field.

- To change the value from Y to N for only select instances, type / on the action lines next to those specific entries.

6 Press F3 to save your changes.

**Renaming an option set**

Use this procedure to rename an existing option set.

1 Type R next to the option set that you want to rename and press Enter.
The Rename Option Set panel (Figure 13 on page 58) is displayed.

**Figure 13: Rename Option Set panel**

```
+--------- Rename Option Set - DBCRIOPT ----------+
|                                                 |
| Enter the option set name and description then  |
| press Enter. Press CANCEL or END to cancel     |
| this action.                                    |
|                                                 |
| Name . . . . DBCRIOPT                           |
| Description . . DOMPLEX FOR DBCRI101            |
|                                                 |
+-------------------------------------------------+
```

2 Enter the new name of the option set and an updated description.

3 Press Enter to save your changes and return to the previous panel.

**Deleting an option set**

Use this procedure to delete an obsolete option set that you no longer need.

1 On the Option Sets panel, type D next to the option set that you want to delete and press Enter.

2 At the Confirm Option Set delete panel (Figure 14 on page 58), press Enter to confirm that you want to delete this option set.

**Figure 14: Confirm Option Set Delete panel**

```
+--------------- Confirm Option Set Delete ---------------+
|                                                         |
| Option set name:                                        |
| DBCRIOPT                                               |
|                                                         |
| Description:                                           |
| DOMPLEX FOR DBCRI101                                    |
|                                                         |
| _ Enter "/" to turn Confirm Option Set Delete off       |
|                                                         |
| Press ENTER to confirm delete.                         |
| Press CANCEL or END to cancel delete.                  |
+--------------------------------------------------------+
```

**Filtering option sets**

Use this procedure to filter values in an option set so that not all options are displayed.
To filter option sets from the menu

1 From the Filter menu, select the filter option that you want to use.

```
<table>
<thead>
<tr>
<th></th>
<th>1. Off</th>
<th>DOMPLEX FOR DBCRI101 Scroll ===&gt; CSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>2.</td>
<td>Required Options</td>
</tr>
<tr>
<td></td>
<td>3.</td>
<td>Modified Options</td>
</tr>
<tr>
<td>Filter</td>
<td>4.</td>
<td>Invalid Options</td>
</tr>
<tr>
<td>+</td>
<td>DOM</td>
<td>5. Default Options</td>
</tr>
<tr>
<td>+</td>
<td>Data</td>
<td>6. Reset Filter</td>
</tr>
<tr>
<td>+</td>
<td>OutGp</td>
<td>Parameters that apply to entire DOMPLEX</td>
</tr>
<tr>
<td>+</td>
<td>DCID</td>
<td>Data Collector(DBC) subsystems in DOMPLEX</td>
</tr>
<tr>
<td>+</td>
<td>DspSize</td>
<td>DB2 Sub-systems to be monitored</td>
</tr>
<tr>
<td>+</td>
<td>(9)</td>
<td>Output Groups - valid range: 001-256</td>
</tr>
</tbody>
</table>
```

You can:

- Display only a particular type of option (required, modified, invalid, or default)

  **Note**
  
  Modified options are those for which the current value differs from the default value that shipped with the product.

- Turn off filtering so that all options are displayed

- Reset filters

2 Press Enter.

To filter option sets by issuing a command

1 At the Command line for the panel, type one of the following commands:

- **FILTOFF**
- **FILTREQ**
- **FILTMOD**
- **FILTINV**
- **FILTDEF**

  **Tip**
  
  For more information, see “LGC interface commands” on page 60.

2 Press Enter.
LGC command reference

LGC includes commands that you can issue through the LGC interface and operator commands that you issue from the console. This section includes descriptions of those commands.

LGC interface commands

You can enter commands in the Command line of the DB2 Product Configuration (LGC) interface.

Table 7 on page 60 lists these commands.

Table 7: Command line commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABOUT</td>
<td>Displays information about LGC including the DBC connected to and applied maintenance for LGC and the components that it uses</td>
</tr>
<tr>
<td>CANCEL</td>
<td>Cancels any changes</td>
</tr>
<tr>
<td>FILTDEF</td>
<td>Shows default options</td>
</tr>
<tr>
<td>FILTINV</td>
<td>Shows invalid options</td>
</tr>
<tr>
<td>FILTMOD</td>
<td>Shows all modified options</td>
</tr>
<tr>
<td>FILTOFF</td>
<td>Turns off filtering and shows all options</td>
</tr>
<tr>
<td>FILTREQ</td>
<td>Shows all required options</td>
</tr>
<tr>
<td>GENHELP</td>
<td>Opens the general Help panel for LGC</td>
</tr>
<tr>
<td>HELP</td>
<td>Opens ISPF Help</td>
</tr>
<tr>
<td>PRODHELP</td>
<td>Opens the Help panel for the BMC product</td>
</tr>
<tr>
<td>SAVE</td>
<td>Saves changes to the current option set</td>
</tr>
<tr>
<td>SAVEAS</td>
<td>Saves changes to a new option set</td>
</tr>
</tbody>
</table>

Some of these commands require you to select an option set before issuing the command (for example, FILTDEF, FILTINV, FILTMOD, FILTOFF, FILTREQ, PRODHELP, SAVE, and SAVEAS).

LGC operator commands

You can enter the LGC operator commands in the console or from SDSF either by:
Issuing the command using the DBC subsystem ID directly

\[ \text{/dbsid LGC,STATUS,SET,CONFIG} \]

Issuing MODIFY command

\[ \text{/F dbsid,LGC,STATUS,SET,CONFIG} \]

Table 8 on page 61 lists the commands that are defined by the LGC datastore manager when it starts and are deleted when it stops. In addition, you can define and use the LGC,START command to start all LGC agents.

Table 8: LGC commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGC,COMMANDS</td>
<td>Displays metrics related to the different commands (mostly internal) that have been processed by the option set manager agent.</td>
</tr>
<tr>
<td>LGC,REFRESH</td>
<td>Loads new copies of LGC modules and stops or starts agents.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You typically use this command after applying an LGC PTF.</td>
</tr>
<tr>
<td>LGC,REGSTAT</td>
<td>Displays the status of the RTCS Subsystem system registry that is currently being used in the LGCDSMPR output DD statement. The same information is displayed when the LGC agents first come up.</td>
</tr>
<tr>
<td>LGC,RELQUES</td>
<td>Releases the internal queues (excluding the sessions queue) used by the option set manager.</td>
</tr>
<tr>
<td>LGC,STATUS</td>
<td>Displays information about option sets that are currently being edited in the LGCOSMPR output DD statement. The displayed information includes:</td>
</tr>
<tr>
<td></td>
<td>- The product name and version</td>
</tr>
<tr>
<td></td>
<td>- The user ID</td>
</tr>
<tr>
<td></td>
<td>- The option set name</td>
</tr>
<tr>
<td></td>
<td>- The option set description</td>
</tr>
<tr>
<td></td>
<td>If no option sets are being edited, a message indicating that is issued.</td>
</tr>
<tr>
<td>LGC,STOP</td>
<td>Stops all LGC agents.</td>
</tr>
<tr>
<td>LGC,STORAGE</td>
<td>Displays information about the amount of storage that is being used by LGC agents. For more information about the STORAGE command, see “LGC,STORAGE command” on page 63.</td>
</tr>
<tr>
<td>LGC,TRACE</td>
<td>Sets the trace control data and enable or disable tracing.</td>
</tr>
<tr>
<td></td>
<td>For more information about the LGC,TRACE command, see “LGC,TRACE command” on page 62.</td>
</tr>
</tbody>
</table>
**LGC,TRACE command**

Use the LGC,TRACE command to set the trace control data and to enable or disable tracing. The command can take the following syntax constructions. Optional parameters are shown in brackets.

**Note**

You must obtain a valid trace key from BMC Customer Support. Do not activate the trace key unless requested to do so by BMC Customer Support.

<table>
<thead>
<tr>
<th>Syntax examples that set trace options</th>
<th>Issue this command</th>
</tr>
</thead>
<tbody>
<tr>
<td>A sequential data set a</td>
<td>LGC,TRACE,SET,DSN=RDAMSM.LGCOPTS</td>
</tr>
<tr>
<td>A sequential data set and save the options in the datastore configuration</td>
<td>LGC,TRACE,SET,DSN=RDAMSM.LGCOPTS,SAVE</td>
</tr>
<tr>
<td>A sequential data set and immediately activate them</td>
<td>LGC,TRACE,SET,DSN=RDAMSM.LGCOPTS,ACTIVATE</td>
</tr>
<tr>
<td>A PDS member</td>
<td>LGC,TRACE,SET,DSN=RDAMSM.PDSV(OPTS)</td>
</tr>
<tr>
<td>a PDS member and save the options in the datastore configuration</td>
<td>LGC,TRACE,SET,DSN=RDAMSM.PDSV(OPTS),SAVE</td>
</tr>
<tr>
<td>A PDS member and immediately activate them</td>
<td>LGC,TRACE,SET,DSN=RDAMSM.PDSV(OPTS),ACTIVATE</td>
</tr>
<tr>
<td>The datastore configuration</td>
<td>LGC,TRACE,SET,CONFIG</td>
</tr>
</tbody>
</table>

If you use the ACTIVATE syntax, you do not have to issue an LGC,TRACE,ON command. If you do not use the ACTIVATE syntax, the trace info will not be written to the OSM agent.

Table 10 on page 63 shows examples of commands that you can issue to enable tracing from the datastore or option set manager.
Table 10: Syntax examples that enable tracing

<table>
<thead>
<tr>
<th>To enable tracing in</th>
<th>Issue this command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only the datastore manager</td>
<td>LGC,TRACE,ON,DSM</td>
</tr>
<tr>
<td>Only option set manager</td>
<td>LGC,TRACE,ON,OSM</td>
</tr>
<tr>
<td>All LGC agents</td>
<td>LGC,TRACE,ON,ALL</td>
</tr>
</tbody>
</table>

Table 11 on page 63 shows examples of commands that you can issue to disable tracing from the datastore or option set manager.

Table 11: Syntax examples that disable tracing

<table>
<thead>
<tr>
<th>To disable tracing in</th>
<th>Issue this command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only the datastore manager</td>
<td>LGC,TRACE,OFF,DSM</td>
</tr>
<tr>
<td>Only the option set manager</td>
<td>LGC,TRACE,OFF,OSM</td>
</tr>
<tr>
<td>All LGC agents</td>
<td>LGC,TRACE,OFF,ALL</td>
</tr>
</tbody>
</table>

**LGC,STORAGE command**

Use the LGC,STORAGE command to display the amount of storage being used by one or more LGC agents. The agents that will process the command are determined by the parameter following the STORAGE verb.

The syntax for this command is as follows:

LGC,STORAGE,[ALL|DSM|OSM]

Table 12 on page 63 shows examples of commands you can issue to display the amount of used storage.

Table 12: Syntax examples for displaying storage

<table>
<thead>
<tr>
<th>To display storage by using</th>
<th>Issue this command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only the datastore manager</td>
<td>LGC,STORAGE,DSM</td>
</tr>
<tr>
<td>Only the option set manager</td>
<td>LGC,STORAGE,OSM</td>
</tr>
<tr>
<td>All LGC agents</td>
<td>LGC,STORAGE,ALL</td>
</tr>
</tbody>
</table>
LGCUTIL batch utility

The LGCUTIL batch utility allows you to pass commands to the LGC component.

All commands are passed to LGCUTIL within the <lgcutil> element (as described in “lgcutil” on page 65).

```xml
<lgcutil version="1010">
  <commandName>
    <commandContent/>
  </commandName>
</lgcutil>
```

Table 13 on page 64 lists the command elements that you can issue through the LGCUTIL utility.

Table 13: Command elements for the LGCUTIL utility

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>Creates a new option set</td>
<td>“create” on page 66</td>
</tr>
<tr>
<td>deregister</td>
<td>Removes an object's registration entry from the LGC datastore</td>
<td>“deregister” on page 66</td>
</tr>
<tr>
<td>export</td>
<td>Exports products, solutions, or option sets to a sequential file</td>
<td>“export” on page 67</td>
</tr>
<tr>
<td>import</td>
<td>Imports option sets or product and solution definitions into the LGC datastore</td>
<td>“import” on page 68</td>
</tr>
<tr>
<td>list</td>
<td>Lists the registration information for one or more products or a DBC SSID or group</td>
<td>“list” on page 69</td>
</tr>
<tr>
<td>migrate</td>
<td>Migrates option sets to a new release</td>
<td>“migrate” on page 69</td>
</tr>
<tr>
<td>register</td>
<td>Registers an object to the LGC datastore</td>
<td>“register” on page 70</td>
</tr>
<tr>
<td>rename</td>
<td>Renames one or more option sets.</td>
<td>“rename” on page 71</td>
</tr>
<tr>
<td>update</td>
<td>Updates option sets.</td>
<td>“update” on page 71</td>
</tr>
<tr>
<td></td>
<td>Note: Do not issue this command unless directed to do so by BMC Support.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: Do not issue this command unless directed to do so by BMC Support</td>
<td></td>
</tr>
</tbody>
</table>

In the description of the elements, the notation following the child elements indicate occurrence information.

- The first number denotes the minimum occurrences of the element (0 indicates that the element is optional).
The second number denotes the maximum occurrences of the element (* means unlimited).

Descriptions of elements that are used within these elements are described in “Supporting elements” on page 72.

Figure 15 on page 65 shows an example of sending a command to LGCUTIL.

**Figure 15: Example of sending a command to LGCUTIL**

```
//LGCCROS  JOB  PAFR,&SYSUID,CLASS=Q,MSGCLASS=X,REGION=OM,
//   NOTIFY=&SYSUID,TIME=1444
//*ROUTE XEQ BMCPLX1
//*JOBPARM SYSAFF=DB2A
//*--------------------------------------------------
//* Example: Create a new option set
//*--------------------------------------------------
//LGC    EXEC  PGM=LGCUTIL
//STEPLIB  DD  DISP=SHR,DSN=LGC.LOAD
//LGCUTLPR DD  SYSOUT=* 
//LGCUTLER DD  SYSOUT=*  
//STOUT  DD  *  
<lgcutil version="1010">
    <create>
        <optionset>
            <prodcode>DOM</prodcode>
            <prodvrsn>1010</prodvrsn>
            <name>MSM$OPT8</name>
            <description>New option set</description>
        </optionset>
    </create>
</lgcutil>
```

**lgcutil**

The required `<lgcutil>` element is the root node of the document that sends commands to LGCUTIL. This element must be the first node in the document. Its immediate children are the command elements that drive the LGCUTIL program.

**Parent element:** Not applicable.

**Child elements:** `<register>` (0:*), `<create>` (0:*), `<deregister>` (0:*), `<import>` (0:*), `export` (0:*), `<list>` (0:*), and `<rename>` (0:*)

**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>string</td>
<td>no</td>
<td>none</td>
<td>Specifies the version number of the LGCUTIL utility.</td>
</tr>
</tbody>
</table>

**Example:**

```
<lgcutil version="1010">
    <list>
        <product>
            <prodcode>AFR</prodcode>
            <prodvrsn>1020</prodvrsn>
        </product>
    </list>
</lgcutil>
```
**create**

The `<create>` command creates a new option set.

**Parent element:** `<lgcutil>`

**Child elements:** `<optionset>` (0:*)

**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duplicate</td>
<td>string</td>
<td>no</td>
<td>error</td>
<td>Specifies how to handle duplicate objects. Valid values are <strong>error</strong> and <strong>warning</strong>. If you specify <strong>warning</strong>, any duplicate object errors are reported as warnings.</td>
</tr>
</tbody>
</table>

**Example:**

```xml
<lgcutil version="1010">
  <create>
    <optionset>
      <prodcode>DOM</prodcode>
      <prodvrsn>1010</prodvrsn>
      <name>MSM$OPT8</name>
      <description>New option set</description>
    </optionset>
  </create>
</lgcutil>
```

**deregister**

Use the `<deregister>` command element to remove an option set, product, or solution from the LGC data store or to deregister a DBC subsystem.

**Parent element:** `<lgcutil>`

**Child elements:** `<dbc>` (0:*), `<defaultdbc>` (0:*), `<optionset>` (0:*), `<solution>` (0:*), and `<product>` (0:*)

**Attributes:** None.

**Example:**

```xml
<lgcutil version="1010">
  <deregister>
    <optionset>
      <prodcode>DOM</prodcode>
      <prodvrsn>1010</prodvrsn>
    </optionset>
  </deregister>
</lgcutil>
```
export

Use the <export> command element to export an object or a set of objects to a sequential file.

The export command can export:

- A single option set
- A group of option sets
- A product with or without its option sets
- A solution definition with or without its products and option sets
- Data store configurations

Output from an <export> command includes an <import> element to identify the imported data as the product of an <export> command. Output from an <export> command can be imported unchanged into another LGC repository.

**Parent element:** <lgcutil>

**Child elements:** <ddname> (1:1), <products> (0:*), <solutions> (0:*), <optionset> (0:*), <optionsets> (0:*), and <config> (0:*)

**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>format</td>
<td>string</td>
<td>no</td>
<td></td>
<td>Valid values are xml, none, and keyword. An export command can create XML as either formatted or unformatted. If the utility creates formatted XML, it includes an XML declaration. Output from the &lt;export&gt; command includes an &lt;import&gt; element. Only a single option set may be exported in keyword format.</td>
</tr>
</tbody>
</table>

**Example:**

```xml
<lgcutil version="1010">
  <export format="xml">
    <products>
      <product>
        <prodvrsn>1010</prodvrsn>
      </product>
    </products>
    <ddname>V1010</ddname>
  </export>
</lgcutil>
```
import

Use the <import> command element to import objects into the LGC datastore:

These objects can include:

- A single option set
- Many option sets
- Products with their defined option sets
- Solutions with their defined option sets and products
- Datastore configurations

Input to the <import> command can be the output from an <export> command.

Parent element: <lgcutil>
Child elements: <ddname> (1:1)
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace</td>
<td>string</td>
<td>no</td>
<td>no</td>
<td>Valid values are yes and no. If you specify yes, any solution definition, product definition, or option set that you import will replace any existing object of the same name.</td>
</tr>
<tr>
<td>duplicate</td>
<td>string</td>
<td>no</td>
<td>error</td>
<td>Valid values are error and warning. If you specify warning, any duplicate object errors are reported as warnings.</td>
</tr>
<tr>
<td>setstats</td>
<td>string</td>
<td>no</td>
<td>no</td>
<td>Valid values are yes and no. If you specify yes, the statistics in each object reflect the date, time, and user ID of the job. Otherwise, the statistics are unchanged from the imported file.</td>
</tr>
</tbody>
</table>

Example:

```xml
<lgcutil version="1010">
  <import replace="yes">
    <ddname>IMPXML</ddname>
  </import>
</lgcutil>
```
list

Use the `<list>` command element to list the registration information for a product, DBC subsystem, or DBC group definition.

**Parent element:** `<lgcutil>`
**Child elements:** `<dbc>` (0:*), `<product>` (0:*), `<products>` (0:*), `<solution>` (0:*), and `<solutions>` (0:*)
**Attributes:** None.
**Example:**

The following example shows a `<list>` command element that includes a single product:

```
<lgcutil version="1010">
 <list>
  <product>
   <prodcode>AFR</prodcode>
   <prodvrsn>1020</prodvrsn>
  </product>
  <dbc ssid="LGQR"/>
  <dbc dbgroup="DBCMSM"/>
 </list>
</lgcutil>
```

The following example shows a `<list>` command element that includes the `<products>` element:

```
<lgcutil version="1010">
 <list>
  <products>
   <product>
    <prodcode>ACP</prodcode>
    <prodvrsn>1020</prodvrsn>
   </product>
  </products>
 </list>
</lgcutil>
```

migrate

Use the `<migrate>` command element to migrate object sets to a new release. Do not issue this command unless directed to do so by BMC Support.

**Parent element:** `<lgcutil>`
**Child elements:** `<optionsets>` (1:1)
**Attributes:**
replace

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace</td>
<td>string</td>
<td>no</td>
<td>no</td>
<td>Specifies whether to replace option sets with the same name. Valid values are yes and no. If you specify yes, the &lt;migrate&gt; command will replace any option set of the same name.</td>
</tr>
<tr>
<td>duplicate</td>
<td>string</td>
<td>No</td>
<td>error</td>
<td>Specifies whether duplicate entries are handled as warnings or errors. Valid values are error and warning. If you specify warning, any duplicate object errors are reported as warnings.</td>
</tr>
<tr>
<td>savewitherrors</td>
<td>string</td>
<td>No</td>
<td>no</td>
<td>Specifies whether the &lt;migrate&gt; command will save an option set even if it contains existing errors or if errors occur in the updates from the LGCUPD DD statement. Only valid updates are saved. Valid values are yes and no. If you specify yes, option sets will be saved with errors.</td>
</tr>
</tbody>
</table>

Example:

```xml
<lgcutil version="1010">
  <migrate>
    <optionsets>
      <product>
        <prodcode>AFR</prodcode>
        <prodvrsn>1020</prodvrsn>
      </product>
      <fromvrsn>1010</fromvrsn>
      <name type="regex">^AFR\$[a-z,A-Z][1,4]\$</name>
    </optionsets>
  </migrate>
</lgcutil>
```

register

Use the <register> command element to add an option set or product to the LGC datastore or to register a DBC subsystem.

Parent element: lgcutil
Child elements: product (0:*), optionset (0:*), dbc (0:*), and defaultdbc (0:*)
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duplicate</td>
<td>string</td>
<td>no</td>
<td>error</td>
<td>Indicates whether to manage duplicate objects as errors or warnings. Valid values are error and warning. If you specify warning, any duplicate object errors are reported as warnings.</td>
</tr>
</tbody>
</table>
rename

Use the <rename> command to rename one or more option sets. An option set with the same name must not already exist. Also, the rename command should not leave any option set instances without a base option set.

**Parent element:** <lgcutil>

**Child elements:** <optionsets> (1:1)

**Attributes:** None.

**Example:**

```xml
<lgcutil version="1010">
  <rename>
    <optionsets>
      <optionset>
        <prodcode>AFR</prodcode>
        <prodvrsn>1020</prodvrsn>
        <name>AFR$OPTS$</name>
        <newname>AFRDB2A</newname>
        <newdescription>This is an option set used on LPAR DB2A</newdescription>
      </optionset>
    </optionsets>
  </rename>
</lgcutil>
```

update

Use the <update> command element to update option values within specified option sets. Do not issue this command unless directed to do so by BMC Support.

**Parent element:** <lgcutil>

**Child elements:** <optionsets> (1:1)

**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>savewitherrors</td>
<td>string</td>
<td>no</td>
<td>no</td>
<td>Indicates whether to save the option set if it contains errors. Valid values are yes and no. If you specify yes, the &lt;update&gt; command saves the option set even if it contains existing errors or if the updates from the LGCUPD DD statement contains errors. Only valid updates are saved.</td>
</tr>
</tbody>
</table>
Supporting elements

Different LGCUTIL command elements include different supporting elements. This section describes the elements and indicates which elements can contain the specified element, any child elements that you can include, attributes of the element, and example of the syntax.

In the description of the elements, the notation following the child elements indicate occurrence information.

- The first number denotes the minimum occurrences of the element (0 indicates that the element is optional).
- The second number denotes the maximum occurrences of the element (* means unlimited).

The following elements are included in this section:

- “config” on page 73
- “dbc” on page 74
- “ddname” on page 74
- “defaultdbc” on page 75
- “description” on page 76
- “dsn” on page 76
- “file” on page 76
- “files” on page 77
- “help” on page 78
- “instance” on page 79
config

The <config> element identifies the LGC configuration data associated with a particular DBC subsystem. You can use the <export> command to export the data to an external file, which then can be imported.

Parent element: <export>
Child elements: None.
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssid</td>
<td>string</td>
<td>no</td>
<td>&lt;connected ssid&gt;</td>
<td>Identifies the name of the DBC subsystem whose configuration is referenced.</td>
</tr>
</tbody>
</table>

Example:

```xml
<export format="xml">
<ddname>XML</ddname>
```
This `<dbc>` element defines a DBC subsystem or group for which to list information or to register or deregister.

**Parent element:** `<register>`, `<deregister>`, or `<list>`  
**Child elements:** `<rtcsreg> (0:1)`  
**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssid</td>
<td>string</td>
<td>no</td>
<td></td>
<td>Defines the subsystem identifier of the DBC subsystem.</td>
</tr>
<tr>
<td>dbcgroup</td>
<td>string</td>
<td>no</td>
<td></td>
<td>Defines the group name of the DBC subsystem.</td>
</tr>
</tbody>
</table>

- When used within a `<list>` command and `default="yes"` is specified, this value acts as a qualifier.
- When used within a `<register>` or `<deregister>` command, this value specifies the name of the DBC group to be registered. The maximum length of this value is 8.

**Example:**

```xml
<register>
  <dbc ssid="DBCM" dbcgroup="DBCMSM">
    <rtcsreg xcfgroup="DBCMGRP">
      <regdsn>LGC.WMSM1011.RTCS.REGISTRY</regdsn>
    </rtcsreg>
  </dbc>
</register>
```

### ddname

The `<ddname>` element refers to a valid JCL DD name in the JCL for LGCUTIL.

- If used within the `<export>` command, the DD name refers to the target location for the export.
- If used within the `<import>` command, the DD name refers to the location of the data to be imported. Input to the `<import>` command can be the output from an `<export>` command from another LGC repository.

**Parent element:** `<import>` or `<export>`  
**Child elements:** None.  
**Attributes:** None.  
**Example:**

```xml
<export format="xml">
  <ddname>DOM</ddname>
</export>
```
The `<defaultdbc>` element identifies the name of the default DBC group for the system. The value of the `<defaultdbc>` element must be a valid DBC group name.

- If used within a `<register>` command, the named group becomes the default group for the system. If a default group already exists, you must include the `replace="yes"` attribute to replace the existing group.

- If used within the `<deregister>` command, the value must correspond to the existing default group name for the command to complete successfully.

**Parent element:** `<register>` or `<deregister>`  
**Child elements:** None.  
**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace</td>
<td>string</td>
<td>no</td>
<td>no</td>
<td>Valid values are <code>yes</code> and <code>no</code>. If you specify <code>yes</code> within a <code>&lt;register&gt;</code> command, the value specified within the element replaces any existing default DBC group name.</td>
</tr>
</tbody>
</table>

**Example:**

The following example illustrates how to use the `<defaultdbc>` element within a `<register>` command:

```
<register>
  <defaultdbc replace="yes">DBCRI101</defaultdbc>
</register>
```

The following example illustrates how to use the `<defaultdbc>` element within a `<deregister>` command:

```
<deregister>
  <defaultdbc>DBCRI101</defaultdbc>
</deregister>
```
description

The <description> element contains a description of the option set. The default length for a description is 30 bytes.

Parent element: <optionset>
Child elements: None.
Attributes: None.
Example:

```xml
<deregister>
  <optionset>
    <prodcode>C01</prodcode>
    <prodvrsn>VC</prodvrsn>
    <name>CO01OPTS</name>
    <description>Option Set for CO01</description>
  </optionset>
</deregister>
```

dsn

This <dsn> element defines a data set name. You use this element in a template definition to define the data set name where the template is stored. It must be a legal data set name.

Parent element: <template> or <help>
Child elements: None.
Attributes: None.
Example:

```xml
<register>
  <product replace="yes" id="ZLGCA10" prodlist="no">
    <prodcode>LGC</prodcode>
    <prodvrsn>1011</prodvrsn>
    <name>DB2 Product Configuration</name>
    <templates>
      <template type="help">
        <dsn>ROHMXN.IVP101MV.BMCXML</dsn>
        <member>LGCHELP</member>
        <help>
          <dsn>ROHMXN.IVP101MV.BMCXML</dsn>
          <member>LGCHELP</member>
        </help>
      </template>
    </templates>
    <files>
      </files>
  </product>
</register>
```

file

The <file> element defines a file name within a product or solution definition or as part of a template definition.

- When the <file> element is part of a product or solution definition, this element is a child of the <files> element and you use it to specify a file used by product or solution exits.
When the <file> element is a child of the <template> element, you use this element to identify a ZFS or HFS file where the template resides.

**Parent element:** <files> or <template>

**Child elements:** None.

**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>no</td>
<td>dsn</td>
<td>Indicates the type of file defined. Valid values are <code>dd</code> and <code>dsn</code>.</td>
</tr>
<tr>
<td>fmt</td>
<td>string</td>
<td>no</td>
<td>load</td>
<td>The only valid value for this attribute is <code>load</code>.</td>
</tr>
<tr>
<td>id</td>
<td>string</td>
<td>no</td>
<td></td>
<td>Identifies a unique string identifier for the file. Use this attribute to refer to the file with the refid attribute on another &lt;file&gt; definition.</td>
</tr>
<tr>
<td>refid</td>
<td>string</td>
<td>no</td>
<td></td>
<td>Identifies the id attribute of another file from which to pick up its characteristics.</td>
</tr>
</tbody>
</table>

**Example:**

```xml
<register>
  <product id="ZLGCA10" prodlst="no" replace="yes">
    <name>BMC Product Management</name>
    <prodcde>LGC</prodcde>
    <prodvrsn>1011</prodvrsn>
    <files>
      <file id="kjdexit">RDAKJD.WKJD.LOAD.EXIT</file>
    </files>
    <templates>
      <template type="help">
        <dsn>LGC.WDBB1011.USERXML</dsn>
        <member>LGCHELP</member>
        <help>
          <dsn>LGC.WDBB1011.XML</dsn>
          <member>LGCHELP</member>
        </help>
      </template>
      <template type="optcoll">
        <dsn>LGC.WDBB1011.USERXML</dsn>
        <member>OPTCOLL</member>
        <help>
          <dsn>LGC.WDBB1011.USERXML</dsn>
          <member>OPTCOLLH</member>
        </help>
      </template>
    </templates>
  </product>
</register>
```

**files**

The <files> element groups a set of <file> elements in the product or solution definition. The set of file names are used by product or solution exits.

**Parent element:** <product> or <solution>

**Child elements:** <file> (0:*)

**Attributes:**
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| hlq   | string | no        |         | The high-level qualifier used to prefix any file names defined within this element.

**Example:**

```xml
<register>
  <product id="ZLGCA10" prodlist="no" replace="yes">
    <name>BMC Product Management</name>
    <prodcode>LGC</prodcode>
    <prodvrsn>1011</prodvrsn>
    <files>
      <file id="kjdexit">RDAKJD.WKJD.LOAD.EXIT</file>
    </files>
    <templates>
      <template type="help">
        <dsn>LGC.WDBB1011.USERXML</dsn>
        <member>LGCHELP</member>
      </template>
      <template type="optcoll">
        <dsn>LGC.WDBB1011.USERXML</dsn>
        <member>OPTCOLL</member>
      </template>
    </templates>
  </product>
</register>
```

**help**

The `<help>` element indicates the location of a help file for a template. This element is used during product registration and indicates where the help file for a template resides.

**Parent element:** `<template>`

**Child elements:** `<dsn>` (1:1) and `<member>` (0:1)

**Attributes:** None.

**Example:**

```xml
<register>
  <product replace="yes" id="ZLGCA10" prodlist="no">
    <prodcode>LGC</prodcode>
    <prodvrsn>1011</prodvrsn>
    <name>DB2 Product Configuration</name>
    <templates>
      <template type="help">
        <dsn>ROHMXN.IVP101MV.BMCXML</dsn>
        <member>LGCHELP</member>
      </template>
    </templates>
  </product>
</register>
```
instance

The `<instance>` element names an 8-character instance of an option set.

Parent element: `<optionset>` or `<optionsets>`
Child elements: None.
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>no</td>
<td></td>
<td>The type of instance string.</td>
</tr>
</tbody>
</table>

Example:

```xml
<optionsets>
  <product>
    <prodcode>AFR</prodcode>
    <prodvrsn>1020</prodvrsn>
  </product>
  <instance>DEDR</instance>
</optionsets>
```

member

The `<member>` element defines a PDS member name. If a template or help file exists in a PDS member, this element is used to indicate the member name. It must be a legal PDS member name.

Parent element: `<template>` or `<help>`
Child elements: None.
Attributes: None.

Example:

```xml
<register>
  <product replace="yes" id="ZLGCA10" prodlist="no">
    <prodcode>LGC</prodcode>
    <prodvrsn>1011</prodvrsn>
    <name>DB2 Product Configuration</name>
    <templates>
      <template type="help">
        <dsn>ROHMXN.IVPCM1MV.BMCXML</dsn>
        <member>LGCHELP</member>
        <help>
          <dsn>ROHMXN.IVPCM1MV.BMCXML</dsn>
          <member>LGCHELP</member>
        </help>
      </template>
    </templates>
    <files/>
    <created>
      <date>20110307</date>
      <time>124732939819</time>
      <userid>RDADAC2</userid>
    </created>
    <updated>
      <date>20110603</date>
      <time>110237504222</time>
      <userid>ROHMXN</userid>
    </updated>
  </product>
</register>
```
name

The `<name>` element contains the name of a product, solution, or option set. For option set names, the name must conform to PDS naming conventions.

**Parent element:** `<product>`, `<solution>`, `<optionset>`, or `<optionsets>`

**Child elements:** None.

**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>no</td>
<td></td>
<td>The type of name string.</td>
</tr>
</tbody>
</table>

**Example:**

```xml
<export format="xml">
  <ddname>DOM</ddname>
  <optionsets>
    <optionset>
      <name>ADS8</name>
      <prodcode>DOM</prodcode>
      <prodvrsn>1010</prodvrsn>
    </optionset>
    <optionset>
      <name>AFDPLEX</name>
      <prodcode>DOM</prodcode>
      <prodvrsn>1010</prodvrsn>
    </optionset>
  </optionsets>
</export>
```

newdescription

The `<newdescription>` element specifies a new description for an option set. Use this element within `<optionset>` elements defined within `<rename>` commands. If the `<newdescription>` element is specified in either of these commands, the command must identify only one option set.

**Parent element:** `<optionset>`

**Child elements:** None.

**Attributes:** None.

**Example:**

```xml
<rename>
  <optionsets>
    <optionset>
      <prodcode>AFR</prodcode>
      <prodvrsn>1020</prodvrsn>
      <name>AFR$OPTS</name>
      <newname>AFR$OPT1</newname>
      <newdescription>Option set OPT1</newdescription>
    </optionset>
  </optionsets>
</rename>
```
newname

The `<newname>` element identifies the name to use for an option set when the `<rename>` command is issued. The option set name must conform to PDS naming conventions.

**Parent element:** `<optionset>`
**Child elements:** None.
**Attributes:** None.
**Example:**

```xml
<rename>
  <optionsets>
    <optionset>
      <prodcode>AFR</prodcode>
      <prodvrsn>1020</prodvrsn>
      <name>AFR$OPTS</name>
      <newname>AFR$OPT1</newname>
    </optionset>
  </optionsets>
</rename>
```

optionset

The `<optionset>` element defines the name of the option set to manage.

The following table describes what the option set specifies within each command.

<table>
<thead>
<tr>
<th>Command</th>
<th>What option set defines</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;register&gt;</code> command a</td>
<td>Specifies an option set defined in an external file to import</td>
</tr>
<tr>
<td><code>&lt;export&gt;</code> command</td>
<td>Specifies the name of an option set to export</td>
</tr>
<tr>
<td><code>&lt;create&gt;</code> command</td>
<td>Specifies the name of the option set to create</td>
</tr>
<tr>
<td><code>&lt;rename&gt;</code> command b</td>
<td>Specifies the name of the option to rename</td>
</tr>
</tbody>
</table>

  a When you use the `<optionset>` element within a `<rename>` command, the `<newname>` and `<instance>` elements are the only valid children.

  b When you use the `<optionset>` element within a `<rename>` command, the `<newdescription>` element is the only valid child.

**Parent element:** `<register>`, `<create>`, `<deregister>`, `<export>`, or `<optionsets>`
**Child elements:** `<name> (0:1)`, `<instance> (0:1)`, `<prodcode> (0:1)`, `<prodvrsn> (0:1)`, `<description> (0:1)`, `<newname> (0:1)`, and `<newdescription> (0:1)`

**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dd</td>
<td>string</td>
<td>no</td>
<td></td>
<td>Identifies the DD statement that contains the option set definition. It is only valid when used with the <code>&lt;register&gt;</code> command element</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>-----------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>replace</td>
<td>string</td>
<td>no</td>
<td>no</td>
<td>Identifies whether to replace an existing option set. Valid values are yes and no.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ If you specify yes when registering an option set, the option set replaces any existing option set with the same name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>■ If you specify yes when creating an option set, the option set replaces any existing option set with the same name.</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>no</td>
<td>display_config</td>
<td>Indicates the type of template to use for the option set.</td>
</tr>
</tbody>
</table>

**Example:**

The following example shows how to use the `<optionset>` element when exporting the option set.

```xml
<export format="xml">
  <ddname>DOM</ddname>
  <optionsets>
    <optionset>
      <name>ADS8</name>
      <prodcode>DOM</prodcode>
      <prodvrsn>1010</prodvrsn>
    </optionset>
    <optionset>
      <name>AFDPLEX</name>
      <prodcode>DOM</prodcode>
      <prodvrsn>1010</prodvrsn>
    </optionset>
  </optionsets>
</export>
```

The following example shows how to use the `<optionset>` element when registering an option set.

```xml
<register>
  <!-- Note that no child elements are required -->
  <!-- because the option set is defined completely -->
  <!-- in an external file. -->
  <optionset dd="OPSET" replace="yes"/>
</register>
```

The following example shows how to use the `<optionset>` element when removing an option set.

```xml
<deregister>
  <optionset>
    <prodcode>DOM</prodcode>
    <prodvrsn>1010</prodvrsn>
    <name>ADS8</name>
  </optionset>
</deregister>
```
**optionsets**

The `<optionsets>` element groups `<optionset>` elements. Use this element to contain option set definitions in the `<export>` command when multiple option sets are exported or to identify option sets using the `<rename>` command.

**Parent element:** `<export>`, or `<rename>`

**Child elements:** `<optionset> (0:*), <name> (0:1), <instance> (0:1), <product> (1:1), and <fromvrsn> (0:1)`

**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>no</td>
<td></td>
<td>Indicates that only option sets of the specified template type are to be selected.</td>
</tr>
</tbody>
</table>

**Example:**

```xml
<export format="xml">
  <ddname>DOM</ddname>
  <optionsets>
    <optionset>
      <name>ADS8</name>
      <prodcode>DOM</prodcode>
      <prodvrsn>1010</prodvrsn>
    </optionset>
    <optionset>
      <name>AFDPLEX</name>
      <prodcode>DOM</prodcode>
      <prodvrsn>1010</prodvrsn>
    </optionset>
  </optionsets>
</export>
```

**prodcode**

The `<prodcode>` element contains the internal BMC product code. Specify a valid three-character product code. If this element is used within an `<export>` command, you can use wildcards (for example, specify `<prodcode>A*</prodcode> to identify all product codes that begin with the letter A). You can include the wildcard only at the beginning of the product code value.

**Parent element:** `<product>`, `<solution>`, or `<optionset>`

**Child elements:** None.

**Attributes:** None.

**Example:**

```xml
<export>
  <ddname>XML</ddname>
  <products>
    <product>
      <prodcode>DOM</prodcode>
      <prodvrsn>1010</prodvrsn>
    </product>
  </products>
</export>
```
product

The `<product>` element defines or selects a product.

The way that you use the element depends upon the command in which you are using it.

- When you use it with the `<register>` command, the `<prodcode>`, `<prodvrsn>`, and `<templates>` elements are required children.

- If you use it within an `<export>` command, the `<product>` element specifies a product to select.
  When you use it to select a product, the `<prodcode>` and `<prodvrsn>` elements become optional. The command supports limited wildcard characters within the `<prodcode>` element (for example, `<prodcode>AF*</prodcode>`).

Parent element: `<register>, <deregister>, <products>, <optionsets>, or <list>
Child elements: `<name> (0:1), <prodcode> (0:1), <prodvrsn> (0:1), <templates> (0:1), and <files> (0:1)
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>no</td>
<td></td>
<td>A 7-character attribute that indicates the FMID for the product. When you register a new product, this value is required.</td>
</tr>
<tr>
<td>prodlist</td>
<td>string</td>
<td>no</td>
<td></td>
<td>Indicates whether a product is to be included in a list of all installed products. Valid values are yes and no.</td>
</tr>
<tr>
<td>list</td>
<td>string</td>
<td>no</td>
<td></td>
<td>Controls whether the product is shown on the list of products when Option 2 is selected from the main menu.</td>
</tr>
<tr>
<td>refid</td>
<td>string</td>
<td>no</td>
<td></td>
<td>A 7-character attribute that is required for a product definition in a solution registration. It refers to the <code>id</code> attribute for the product in its definition.</td>
</tr>
<tr>
<td>replace</td>
<td>string</td>
<td>no</td>
<td>no</td>
<td>Indicates whether to replace an existing product. Valid values are <code>yes</code> and <code>no</code>. If you specify <code>yes</code> within the <code>&lt;register&gt;</code> command, the registered product replaces any existing product of the same prodcode/version.</td>
</tr>
</tbody>
</table>

Example:

The following example illustrates the use of a `<prodcode>` element within an export command:

```xml
<export>
  <ddname>XML</ddname>
  <!-- select all products for version 1010 -->
  <products>
    <product>
```
The following example illustrates the use of a wildcard character within the <prodcode> element:

```xml
<export>
  <ddname>XML</ddname>
  <!-- select all products with prodcode beginning -->
  <!-- with 'A' but only for version 1020 -->
  <products>
    <product>
      <prodcode>A*</prodcode>
      <prodvrsn>1020</prodvrsn>
    </product>
  </products>
</export>
```

The following example illustrates a <prodcode> element within the <register> command:

```xml
<register>
  <!-- register a product -->
  <product replace="yes" id="ZLGCA10" prodlist="no">
    <prodcode>LGC</prodcode>
    <prodvrsn>1011</prodvrsn>
    <name>DB2 Product Configuration</name>
    <templates>
      <template type="help">
        <dsn>ROHMXN.IP101MV.BMCXML</dsn>
        <member>LGCHELP</member>
        <help>
          <dsn>ROHMXN.IP101MV.BMCXML</dsn>
          <member>LGCHELP</member>
        </help>
      </template>
    </templates>
    <files>
    </files>
    <created>
      <date>20110307</date>
      <time>124732939819</time>
      <userid>RDADAC2</userid>
    </created>
    <updated>
      <date>20110603</date>
      <time>110237504222</time>
      <userid>ROHMXN</userid>
    </updated>
  </product>
</register>
```

**products**

The <products> element groups a set of products.

Use the <products> element in the <solution> element and in the <export> command element. It can either define all products contained in a solution or specify which products are to be exported. If only a subset of product definitions for a solution are to be exported, specify `products="no"` on the <solution> element and then specify each product to be exported explicitly.
Parent element: <export>, <solution>, or <list>
Child elements: <product> (0:*)
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>string</td>
<td>no</td>
<td>no</td>
<td>Indicates whether to export all product definitions when included within an &lt;export&gt; command. Valid values are yes and no. If you specify yes, the utility exports all product definitions.</td>
</tr>
<tr>
<td>optionsets</td>
<td>string</td>
<td>no</td>
<td>yes</td>
<td>Indicates whether to process option sets when included in the &lt;export&gt; and &lt;list&gt; commands. Valid values are yes and no. If you specify no, option set definitions will not be processed.</td>
</tr>
<tr>
<td>insolution</td>
<td>string</td>
<td>no</td>
<td>yes</td>
<td>Indicates whether to export products that exist in a solution when using an &lt;export&gt; command. Valid values are yes and no. If you specify no, the utility will not export any product that exists in a solution.</td>
</tr>
</tbody>
</table>

Example:

```xml
<export format="xml">
  <products>
    <product>
      <prodvrsn>1010</prodvrsn>
    </product>
  </products>
  <ddname>V1010</ddname>
</export>
```

**prodvrsn**

The <prodvrsn> element defines the version of the product. Specify the product version in `vvrm` format, where `v` is the version, `r` is the release, and `m` is the modification level (for example, 1010).

Parent element: <product>, <solution>, or <optionset>
Child elements: None.
Attributes: None.
Example:

```xml
<export format="xml">
  <products>
    <product>
      <prodvrsn>1010</prodvrsn>
    </product>
  </products>
  <ddname>V1010</ddname>
</export>
```
regdsn

The <regdsn> element defines the RTCS Subsystem system registry data set name. If an instance of LGC uses a private RTCS-managed registry, this element names the linear VSAM data set where the private registry resides.

**Note**
The RTCS Subsystem system registry data set is unique to the DBC group and cannot be shared among DBC groups.

**Parent element:** <rtcsreg>
**Child elements:** None.
**Attributes:** None.
**Example:**

```xml
<register>
  <dbc ssid="DBCM" dbcgroup="DBCMSM">
    <rtcsreg xcfgroup="DBCMGRP">
      <regdsn>LGC.WMSM1011.RTCS.REGISTRY</regdsn>
    </rtcsreg>
  </dbc>
</register>
```

rtcsreg

The <rtcsreg> element defines a private RTCS-managed registry entry for a DBC subsystem. If the private registry needs to be accessed from multiple LPARs, you must specify an XCF group name. If you do not specify an XCF group name, the private registry can only be accessed from one LPAR. Attempting to start the LGC agents in a second DBC in the same DBC group will fail if the private registry is not shared.

**Parent element:** <dbc>
**Child elements:** <regdsn> (0:1)
**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xcfgroup</td>
<td>string</td>
<td>No</td>
<td></td>
<td>Defines an XCF group name for the private RTCS-managed registry. This value must conform to the guidelines for XCF group names. XCF member names are automatically created from the DBC STC name and job number. (for example, LGCTSA1064426). The maximum length of this value is 8. <strong>Note:</strong> The XCF group name must not match what is specified in the REGISTRY-XCF-GROUP parameter defined in member OSZINI** in SYS1.PARMLIB.</td>
</tr>
</tbody>
</table>

**Example:**

```xml
<register>
  <dbc ssid="DBCM" dbcgroup="DBCMSM">
    <rtcsreg>
      <regdsn>LGC.WMSM1011.RTCS.REGISTRY</regdsn>
    </rtcsreg>
  </dbc>
</register>
```
solution

The <solution> element defines or selects a solution. The elements that you include within the <solution> element depend on the command that you are specifying.

- If you are using the <solution> element within a <register> command, you must specify the following required elements:
  
  — <name>
  — <prodcode>
  — <prodvrsn>
  — <templates>

- If you using the <solution> element within a <deregister> or <export> command, you need to specify only the <prodcode> and <prodvrsn> child elements.

Parent element: <register>, <deregister>, <solutions>, or <list>
Child elements: <name> (0:1), <prodcode> (0:1), <prodvrsn> (0:1), <products> (0:1), <templates> (0:1), and <files> (0:1)
Attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>no</td>
<td></td>
<td>A 7-character attribute that defines the FMID. This value is required when registering a solution.</td>
</tr>
<tr>
<td>prodlist</td>
<td>string</td>
<td>no</td>
<td></td>
<td>Indicates whether the solution is included in reports or displays of all installed solutions.</td>
</tr>
<tr>
<td>replace</td>
<td>string</td>
<td>no</td>
<td>no</td>
<td>Indicates whether to replace existing solutions with the same product code and version when using the &lt;register&gt; command. Valid values are yes and no.</td>
</tr>
</tbody>
</table>

Example:

The following example shows how to use the <solution> element to register a solution:

```
<register>
  <solution id="ZRMDA10" prodlist="yes" replace="no"/>
```
The following example shows how to use the `<solution>` element to deregister a product:

```xml
<deregister>
  <solution products="yes">
    <prodcode>RMD</prodcode>
    <prodvrsn>1010</prodvrsn>
  </solution>
</deregister>
```

## solutions

The `<solutions>` element selects a group of solutions within a command.

**Parent element:** `<export>` or `<list>`

**Child elements:** `<solution>` (0:*)

**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>string</td>
<td>no</td>
<td>no</td>
<td>Indicates whether to include all solutions. Valid values are yes and no. If you specify yes, the process will include all solutions.</td>
</tr>
<tr>
<td>optionsets</td>
<td>string</td>
<td>no</td>
<td>yes</td>
<td>Indicates whether to process solutions without their option sets. Valid values are yes and no. If you specify no, solutions are processed without their option sets.</td>
</tr>
<tr>
<td>products</td>
<td>string</td>
<td>no</td>
<td>yes</td>
<td>Indicates whether to list solutions without their associated product definitions. Valid values are yes and no. If you specify no, solutions are exported or listed without the constituent product definitions. To export or list only a subset of product definitions for the solution, specify <code>products=&quot;no&quot;</code> and then specify a <code>&lt;product&gt;</code> element for each desired product definition.</td>
</tr>
</tbody>
</table>
Example:

```
<export format="xml">
  <solutions all="no" optionsets="yes" products="yes"/>
  <ddname>ARM</ddname>
</export>
```

template

The `<template>` element defines a template. This element can define several template types, specified by the `type` attribute.

**Parent element:** `<templates>`  
**Child elements:** `<dsn>` (0:1), `<member>` (0:1), `<help>` (0:1), and `<file>` (0:1)

**Attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| type   | string  | no        | display_config | Identifies the type of the template.  
The maximum length of this value is 16.  
The name must conform to PDS member characteristics, except that it can include the underscore character. |

Example:

```
<register>
  <product replace="yes" id="ZLGCA10" prodlist="no">
  <prodcode>LGC</prodcode>
  <prodvrsn>1011</prodvrsn>
  <name>DB2 Product Configuration</name>
  <templates>
    <template type="help">
      <dsn>ROHMXN.IVP101MV.BMCXML</dsn>
      <member>LGCHELP</member>
      <help>
        <dsn>ROHMXN.IVP101MV.BMCXML</dsn>
        <member>LGCHELP</member>
      </help>
    </template>
  </templates>
  </product>
</register>
```

templates

The `<templates>` element groups a set of templates. If you define a template, the `<template>` element must exist as a child of the `<templates>` element.
Parent element: `<product>` or `<solution>`
Child elements: `<template>` (0:*)
Attributes: None.
Example:

```xml
<register>
  <product replace="yes" id="ZLGCA10" prodlist="no">
    <prodcode>LGC</prodcode>
    <prodvrsn>1011</prodvrsn>
    <name>DB2 Product Configuration</name>
    <templates>
      <template type="help">
        <dsn>ROHMNX.IVP101MV.BMCXML</dsn>
        <member>LGCHELP</member>
      </template>
    </templates>
    <files />
    <created>
      <date>20110307</date>
      <time>124732939819</time>
      <userid>RDADAC2</userid>
    </created>
    <updated>
      <date>20110603</date>
      <time>110237504222</time>
      <userid>ROHMNX</userid>
    </updated>
  </product>
</register>
```
Working with the Next Generation Logger (NGL)

This chapter provides instructions for working with the Next Generation Logger (NGL) technology.

Next Generation Logger (NGL) is a technology that manages logging and retrieval functions including allocating, initializing, and managing log files. NGL minimizes the cost of logging and the potential for resource contention.

NGL uses a file system maintained on z/OS linear data sets for recording captured data into its log files. The NGL uses a "Round Robin" system in which the most current records eventually start replacing the least current records. The number of linear data sets comprising a file system is configurable by the user. When a linear data set is exhausted during insert processing, a switch is made to the next available data set. If all data sets in the file system have been used, the oldest data set is selected as the current logging data set and insertion continues.

Customizing the NGLARCH started task

NGL provides a process to archive the data written to the log files. A sample PROC, NGLARCH, is delivered in the SAMP data set and might need to be customized and copied to your SYS1.PROCLIB (or equivalent) started task library (depending on the products installed).

The NGLARCH started task is spawned by the DBC started task when an archive is requested. DBC spawns NGLARCH when one of the following conditions occur:

- A log file is full.
- SWITCH command is issued.

The definition for the NGL agent identifies the name of the NGLARCH started task.
Currently only the DOM agent uses the NGLARCH started task.

To update or change the name in the agent definition

1. Edit the NGLINIT step from the $465INIT installation job.

2. Locate the <PROCNAME> element:

   `<PROCNAME>NGLARCH</PROCNAME>`

3. Change the name to match the member name in your system PROCLIB and rerun that step.

   Execute the `NGL,REFRESH,piid` command, where `piid` indicates the product instance identifier. For more information about NGL commands, see “NGL commands” on page 100.

   **Note**

   If the DOM agent is active, issue the `DOM,STOP` command before issuing the `NGL,REFRESH` command. Once NGL has started, issue the `DOM,START` command.

Figure 16 on page 94 shows an example of this started task.

**Figure 16: NGLARCH started task**

```
//NGLARCH   PROC
//NGLARCH   EXEC   PGM=NGL9ARCH,
//   ACCT=????,REGION=0M
//STEPLIB   DD DISP=SHR,DSN=?BMC_HLQ?.NGLLINK
//         DD DISP=SHR,DSN=?BMC_HLQ?.DBCLINK
//         DD DISP=SHR,DSN=?BMC_HLQ?.DOMLINK
//SYSPRINT   DD   SYSOUT=*
//REPORT     DD   SYSOUT=*
//NGL$DUMP   DD   DUMMY
```

Table 14 on page 94 describes the DD statements included in the NGL started task.

**Table 14: DD statements for NGLARCH started task**

<table>
<thead>
<tr>
<th>DD statement</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPLIB</td>
<td>The location for NGL, DBC, and DOM load libraries. Depending on the options chosen during the install, the NGL, DBC, and DOM load libraries might be three separate libraries or they might be all in the same library.</td>
</tr>
<tr>
<td>SYSPRINT</td>
<td>Identifies the data set to write potential non-DBC product application messages.</td>
</tr>
<tr>
<td>REPORT</td>
<td>Identifies the data set that contains the output of the NGLARCH process.</td>
</tr>
</tbody>
</table>
Support for multiple NGL registries

By using multiple registries, different products can use a separate registry for its NGL data. For example, MainView Transaction Analyzer can have one NGL registry, while MainView for Java Environments can have its own NGL registry and BMC System Performance for DB2 can use a unique NGL registry independent of the other products. NGL also supports sysplex sharing of the NGL registry that enables NGL instances on separate or the same LPAR to use the same registry.

The NGL registry definition (REGISTRY DD) in the DBC started task procedure is optional. Support is based on defining the target NGL registry using the RTCS Indirect Registry Definition facility (IRDF). By using IRDF, you can define the NGL registry name to RTCS and RTCS will retain that information in the RTCS system registry. NGL agent initialization, based on the start parameters for the agent, determines the registry configuration and in conjunction with RTCS, allocates the proper registry for that agent.

Default behavior during installation

The default behavior of the Installation System at the ZNGLA30 FMID level is to prompt the installer for the NGL registry name and construct the necessary jobs that define the registry if required. The DBC started task procedure will include a REGISTRY DD statement, and the <STARTAGENT> <PARMS> for the NGL agent statement will specify PR=Y. This behavior is required to support a mixed environment of pre-ZNGLA30 FMIDs with ZNGLA30 FMID agents.

To implement the ZNGLA30 enhanced registry support after installation:

1. Verify that the <LOADLIB> statements for the NGR product specify the load libraries from the ZNGLA30 installation.

2. Follow the Implementing multiple NGL registries instructions (see “Implementing multiple NGL registries” on page 96) to customize the required NGL agents for the enhanced registry support.

3. If all NGL agents are at the ZNGLA30 FMID level and using the enhanced registry support, the //REGISTRY DD can be removed from the DBC started task procedure.

<table>
<thead>
<tr>
<th>DD statement</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NGL$DUMP</td>
<td>A dummy statement to be used if a dump has been requested by BMC Support.</td>
</tr>
</tbody>
</table>
Installation into an existing environment

ZNGLA30 operates in compatibility mode with ZNGLA20 or earlier. If you wish to exploit the new NGL registry features provided by ZNGLA30 and ZNGLA30 is being installed into an existing DBC environment with products that might be running with ZNGLA20 or earlier, the NGL registry agent, NGR, must be upgraded to the ZNGLA30 level. NGR at the ZNGLA30 level is downward compatible and supports NGL registry features for ZNGLA20 and earlier releases. Any NGL agent can continue to be at the ZNGLA20 or earlier level using the NGR registry agent at ZNGLA30 until the products using that NGL agent are upgraded. The NGR agent at ZNGLA20 and earlier do not support the new registry features for ZNGLA30.

Implementing multiple NGL registries

Use the following procedure to implement the use of multiple NGL registries:

1. Determine the NGL registry environment you want.
   
   That is, decide how the NGL registries are to be used. Consider the following questions:
   
   - Do you want data for multiple products to share the same NGL registry?
   - Do one or more products require or want their data separated from other product data?
   - Does the NGL registry need to be shared with other NGL agents?

2. Configure the NGL agent parameter (<PARMS> in the <STARTAGENT> definition) to match the desired NGL registry environment.

   The following table describes how the <PARMS> settings determine registry usage:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR</td>
<td>Defines the RTCS registry instance type. This parameter is optional. To define a private registry, use PR=Y. The setting is ignored if the RG parameter is included. If PR=Y is omitted, a private registry is assumed and RD or RG must be included. <strong>Note:</strong> PR=N is not valid. It was previously used for RTCS system registry support.</td>
</tr>
</tbody>
</table>
Parameter | Description
---|---
RD | Defines the RTCS private registry name. This parameter is optional. To define a private registry data set name, use PR=Y and RD=<privateRegistryDatasetName>.  
- If PR=Y is omitted, RD or RG must be included.  
- If PR=Y and RD=<privateRegistryDatasetName> are omitted, NGL scans the TIOT for a REGISTRY DD statement and verifies the DD is present. If the REGISTRY DD statement is missing, the registry open fails.  
- If PR=Y and RD=<privateRegistryDatasetName> are omitted, the RTCS IRDF is used to define the NGL registry data set name.  
- If PR=Y is included and RD is not included, NGL requires the REGISTRY DD statement in the DBC started task procedure.  
- The RD and RG parameters are mutually exclusive.

RG | Defines the NGL registry group. This parameter is optional. To define a registry group, use RG=<groupName>. The name must be 1 to 8 characters.  
- To allow sysplex sharing of the NGL registry, include the RG parameter.  
- The RTCS IRDF will be used to define the NGL registry data set name.  
- The RG and RD parameters are mutually exclusive.  
- This XCF group cannot be shared with other XCF groups such as the LGC, DBC, or DOMPLEX names. The XCF group must be the same for all NGL agents that share a common NGL registry.

3 Configure the RTCS IRDF job to match the desired NGL registry environment.

Sample JCL to run the RTCS IRDF is included in the HQL.XXSAMP data set, member NGLRIRDF. This JCL must be configured to specify the required parameters for the target DBC, NGL, and registry names. A sample of the JCL is shown below. See the "JCL example scenarios" section for additional scenarios.
This RIU input stream illustrates how to customize Indirect Registry Definition facility (IRD) VALUES.

The RIU SYSIN input stream consists of all 80 bytes of the record and columns 73 through 80 are NOT interpreted as sequence numbers.

Set the following RIU OSZRGIRD XML IMPORT file (member) variables according to product customization instructions. They specify the logical private product Registry that is to be indirectly defined.

NGL.DBCID             the target DBC SSID
NGL.NGLID             the target NGL PIID
NGL.DSNAME            the name of the NGL RTCS registry for this NGL instance
NGL.SHARED            if registry is to be sysplex shared, "Y" else code "N"
NGL.NGLPLEX           if registry is to be sysplex shared, then NGLPLEX must be a unique 8 character name to be used for the NGL registry XCF group
NGL.REGID             if registry is to be sysplex shared, then REGID should be the DBCID concatenated with the NGLPLEX name. For example, DBCANGLPX01 for DBCID=DBCA and NGLPLEX=NGLPX01
                                    if registry is not to be shared, then REGID should be the DBCID concatenated with the NGLID. For example, DBCANGL1 for DBCID=DBCA and NGLID=NGL1
                                    * SET NGL.SHARED=
                                    * SET NGL.NGLPLEX=
                                    * SET NGL.REGID=DC02DMA2
                                    * SET NGL.COMMENT=NGL_Private_Registry
                                    * Registry Definition
                                    SET NGL.DBCID=NGA3
                                    SET NGL.NGLID=NGA3
                                    SET NGL.DSNAME=ROHETH.NGA3.NGA3.REGISTRY
                                    SET NGL.SHARED=Y
                                    SET NGL.NGLPLEX=NGR00001
                                    SET NGL.REGID=NGA3NGR00001
                                    SET NGL.COMMENT=NGL_Private_Registry_NGR00001
                                    * Import to RTCS system registry
                                    */

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NGL.DBCID</td>
<td>4 characters</td>
<td>Target DBC SSID of the hosting DBC.</td>
</tr>
<tr>
<td>NGL.NGLID</td>
<td>4 characters</td>
<td>Target NGL PIID of the hosting NGL agent.</td>
</tr>
<tr>
<td>NGL.DSNAME</td>
<td>1 to 44 characters</td>
<td>Name of the NGL registry.</td>
</tr>
<tr>
<td>NGL.SHARED</td>
<td>Y or N</td>
<td>If the registry is to be sysplex shared, specify Y otherwise specify N.</td>
</tr>
</tbody>
</table>
### JCL example scenarios for NGL registries

These example scenarios illustrate how to create the JCL for different RTCS IRDF jobs to match a desired NGL registry environment.

#### Shared registry example

In this example, a registry is to be shared with NGL NGA3 and NGD3. A RTCS IRDF will need to be run on the target LPAR that host NGA3 and a RTCS IRDF will need to be run on the LPAR that host NGD3. Note that the LPAR for NGD3 can be the same as for NGA3. The DBC hosting each NGL can be the same or different. The `<STARTAGENT> <PARMS>` statement should be NGLID=nglPIID,RG=NGR00001 in each NGL definition for NGA3 and NGD3. The required input to the IRDF for each LPAR is below:

For NGA3:

```plaintext
SET NGL.DBCID=NGA3
SET NGL.NGLID=NGA3
SET NGL.DSNAME=ROHETH.NGA3.NGA3.REGISTRY
SET NGL.SHARED=Y
SET NGL.NGLPLEX=NGR00001
SET NGL.REGID=NGA3NGR00001
SET NGL.COMMENT=NGL_Private_Registry_NGR00001
```

For NGD3:

```plaintext
SET NGL.DBCID=NGD3
SET NGL.NGLID=NGD3
SET NGL.DSNAME=ROHETH.NGA3.NGA3.REGISTRY
SET NGL.SHARED=Y
SET NGL.NGLPLEX=NGR00001
SET NGL.REGID=NGD3NGR00001
SET NGL.COMMENT=NGL_Private_Registry_NGR00001
```

The `<STARTAGENT> <PARMS>` would be:

- `<PARMS>NGLID=NGA3,RG=NGR00001</PARMS>` for NGA3
- `<PARMS>NGLID=NGD3,RG=NGR00001</PARMS>` for NGD3
Private registry example

In this example, a private registry is defined. RTCS IRDF will be used for a single NGL instance on a single LPAR. Note that the REGISTRY DD is not required and should be removed from the DBC JCL if present. The <STARTAGENT> <PARMS> coding should be NGLID=nglPIID with no PR, RD, or RG parameter in the NGL definition for DMA2.

```
SET NGL.DBCID=\DC02\n
SET NGL.NGLID=\DMA2\n
SET NGL.DSNAME=\ROHETH.NGA2.NGC3.REGISTRY\n
SET NGL.SHARED=\N\n
SET NGL.NGLPLEX=\n
SET NGL.REGID=\DC02DMA2\n
SET NGL.COMMENT=\NGL_Private_Registry\n```

An alternative to the IRDF coding would be to not define the registry to IRDF and change the <STARTAGENT> <PARMS> to NGLID=DMA2,RD=ROHETH.NGA2.NGC3.REGISTRY.

NGL commands

NGL supports console commands to stop, start, or refresh the NGL agents for a specific NGL product instance identifier (PIID).

Table 15 on page 100 lists the commands that are available for the NGL. In this table, the `dbcssid` value indicates the DBC subsystem ID under which the NGL is running, and the `piid` value indicates the NGL PIID.

**Table 15: NGL commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dbcssid NGL,STOP,piid</td>
<td>Stops the NGL agent for the specified <code>piid</code></td>
</tr>
<tr>
<td>/dbcssid NGL,START,piid</td>
<td>Starts the NGL agent for the specified <code>piid</code></td>
</tr>
<tr>
<td>/dbcssid NGL,REFRESH,piid</td>
<td>Stops and then starts the NGL agent for the specified <code>piid</code></td>
</tr>
<tr>
<td>/dbcssid NGL,DUMP,piid</td>
<td>Takes a dump of the NGL address space and any connected client address spaces Use when BMC Support directs you to do so.</td>
</tr>
<tr>
<td>/dbcssid NGL,CMD,piid,LIST LOGSET {logset_name}</td>
<td>Lists all active logsets and provides status information such as logfile time span and record count information. If <code>logset_name</code> is omitted, all logsets are displayed.</td>
</tr>
</tbody>
</table>

BMC Infrastructure Components Administration Guide
Some products include options that you can use to configure NGL for use with that product. For additional information about using NGL with a particular product or solution, see your production documentation.

### Reporting logset information

You can execute a REXX Exec to display the time span of data of each log file in a specific output group.

The syntax is:

```
NGLBATCH dbc-id ngl-id command logset_name
```

In this syntax:

- `dbc-id` is the DBC subsystem ID
- `ngl-id` is the piid of the NGL agent
- `command` can be LIST LOGSET or STATUS LOGSET
- `logset_name` follows the template of the `dbc-id`, followed by the letter L followed by the output group number

For example: the `logset_name` of Data Collector DC01 and output group 001 is `DC01L001`.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/dbcssid NGL.CMD,piid,DISPLAY LOGSET logset_name</code></td>
<td>Displays the defined logset characteristics to NGL. <em>logset_name</em> is required. <strong>Note:</strong> For DOM users, this information should match the output group definition in the optionset.</td>
</tr>
<tr>
<td><code>/dbcssid NGL.CMD,piid,STATUS LOGSET {logset_name}</code></td>
<td>Displays insert and retrieval statistics for a logset. <em>logset_name</em> is optional. If <em>logset_name</em> is omitted, all logsets are displayed. <strong>Note:</strong> For information on the Data Collector STATUS command, see STATUS on page 102</td>
</tr>
<tr>
<td><code>/dbcssid NGL.CMD,piid,ARCHIVE LOGSET logset_name {logfile_name}</code></td>
<td>Use ARCHIVE with <em>logfile_name</em> to archive a logfile if NGLARCH has failed and the logfile requires another archive. <em>logset_name</em> is required. <em>logfile_name</em> is optional. If <em>logfile_name</em> is included in the ARCHIVE command, the named logfile is archived.</td>
</tr>
</tbody>
</table>
Example

The following example issues a LIST LOGSET command for logset DC01 to DBC server DBCC and NGL PIID NGL1.

```plaintext
NGLBATCH DC01 NGL1 LIST LOGSET DC01L001
```

A REXX EXEC and the JCL to execute it are contained in the SAMP library (XXSAMP or NGLSAMP) in members NGLBATCH and NGLBCMD respectively.

**Figure 17: Sample JCL to execute REXX Exec**

```plaintext
//NGLBATCH JOB ...
//TSOB EXEC PGM=IKJEFT01,DYNAMNBR=10
//SYSPRINT DD SYSOUT=*
//SYSTERM DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSPROC DD DISP=SHR,DSN=your.NGL.SAMP NGL SAMP library
//SYSTSIN DD *
%NGLBATCH DC01 NGL1 LIST LOGSET DC01L001
```

In this syntax:

- DC01 is the DBC subsystem ID
- NGL1 is the NGL piid
- DC01L001 is the DBC subsystem ID (DC01) and 001 is the logset number.

**Figure 18: Example of output**

```plaintext
----- Output for command: LIST LOGSET DC01L001 -----
NGL MESSAGE RC: 00000000  RSN: FFFFFFFF
BMCNGL59704I RESPONSE FROM JOB:DBCPROC2  NGL:NGL1  DBCID:DC01  ASID:0X0178
BMCNGL59695I LOGSET STATUS DSNAME
BMCNGL59696I START TIME ENDTIME
BMCNGL59701I RECORDS DATA BYTES COMP BYTES PERCENT
BMCNGL59697I DC01L001 ACT/INU-C NGL.GRP1.DC01.NGLC.C0000003
BMCNGL59698I DC01L001 ACTIVE -C NGL.GRP1.DC01.NGLC.C0000002
  20120517 12:28:54.442  20120517 12:58:28.938 393155 152.15MB 37.27MB 24
BMCNGL59701I DC01L001 ACTIVE -C NGL.GRP1.DC01.NGLC.C0000001
  20120517 12:28:54.442  20120517 12:58:28.938 393155 152.15MB 37.27MB 24
READY
END
```

**STATUS**

The Data Collector STATUS command returns the status of active output groups. The command can be issued only from the system operator console.
The minimum command entry is \texttt{STATUS} which returns the status on each active output group. The additional parameters allow \texttt{STATUS} to provide status on a particular output group and also list statistics for all or a particular output group.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{DCssid}</td>
<td>Data Collector ID</td>
</tr>
<tr>
<td>\texttt{LOGSET}</td>
<td>\texttt{LOGSET} returns read and write statistics for an output group</td>
</tr>
<tr>
<td>\texttt{LOGSTAT}</td>
<td>\texttt{LOGSTAT} returns a one line status of an output group</td>
</tr>
<tr>
<td>Output_group_name</td>
<td>Specifies which output group details to be displayed</td>
</tr>
<tr>
<td></td>
<td>The output_group_name is in following template: \texttt{DCssid} followed by the letter \texttt{L} followed by the output group number.</td>
</tr>
<tr>
<td></td>
<td>For example: \texttt{DC01L001} for data collector DC01 and output group 001</td>
</tr>
</tbody>
</table>

\textbf{Figure 19: SYNTAX of the STATUS command}

\textbf{Example}

- \texttt{DC01 STATUS}
  - Returns the output group status for all output groups

- \texttt{DC01 STATUS DC01L001}
  - Returns the status for output group 001 from data collector DC01

- \texttt{DC01 STATUS LOGSTAT}
  - Returns the status for all output groups (same as \texttt{STATUS} without parameters)

- \texttt{DC01 STATUS LOGSET}
  - Returns statistics for all output groups

- \texttt{SDC01 STATUS LOGSTAT DC01L001}
  - Returns the status for output group 001 from data collector DC01

- \texttt{DC01 STATUS LOGSET DC01L001}
  - Returns statistics for output group 001 from data collector DC01
Deploying products that use the infrastructure components

This chapter explains how to deploy the following infrastructure components:

- DB2 Component Services (DBC)
- DB2 Product Configuration (LGC)
- Next Generation Logger (NGL)
- Runtime Component System (RTCS)

Carefully review the information in this document to help you plan how to install and deploy your products across your environment.

Overview of deployment

Deploying products that use the infrastructure components requires careful planning. This overview illustrates how you might combine these components into different configurations across LPARs. Although each user's environment is unique, this document attempts to cover a variety of typical situations.

Some users already have their own procedures; these users simply want to know what additional steps are required to set up the new components for different environments.

Requirements for each LPAR

Each LPAR must have the following components (depending on the product installed):

- RTCS started task
- DBC started task
- *(optional)* NGLARCH started task
- BMC Workbench (GUD), BMC Subsystem Optimizer *for zEnterprise* (BRD), LGC, NGL, and DOM agents defined and running under the DBC subsystem
- NGL private or shared registry data set
- LGC private registry data set, which can be shared across multiple DBC subsystems within the same DBC group
- DBC repository data set
- *(Pool Advisor only)* DOMVARS and PMDHIST data sets

You can combine these components into different configurations across LPARs, depending on whether you are sharing DBC groups, RTCS system registries, and LGC private registries.

**Examples of possible deployment environments**

*Figure 20 on page 107* illustrates an environment in which different LPARs have different RTCS registries but share the LGC private registry and the DBC group name. In this example, both LPARs share:

- Runtime data sets
- LGC registry
- DBC group
- Product option sets
The DOM, BRD, and LGC agents and the DBC subsystem communicate across the coupling facility (XCF).

Figure 21 on page 108 illustrates an environment in which different LPARs share:

- RTCS registries
- LGC private registry
- DBC group name
- Product option sets

This example is similar to the previous one but demonstrates that the same setup can be achieved whether the RTCS system registry is shared or unshared.
Figure 22 on page 109 illustrates an environment in which different LPARs have:

- Different RTCS registries
- Different LGC private registries
- Different DBC group names
- Different product option sets

The configuration in this example enables you to keep TEST separate from PROD so that the LGC, DOM, and DBC do not communicate across the coupling facility.
Because the example shown in Figure 22 on page 109 uses a separate LGC private registry on each LPAR, the names of the DOM and GUD option sets might be the same on each LPAR but will be different instances.

Choosing a configuration

The configuration that best suits your environment depends on:

- Whether RTCS is already configured (and if so, whether it is configured as shared or non-shared)
- How many DBC groups you want to have in a sysplex
- Whether you need to have more than one DBC started tasks running on the same LPAR
Preparing for installation

Reviewing this section before starting the installation will help you plan for deployment.

Verifying that the required RTCS level is installed

Each LPAR will have only one RTCS started task running. If you already have RTCS installed and running, use the following procedure to ensure that you are using the required minimum level of RTCS.

**Note**

If you are installing RTCS for the first time, BMC recommends obtaining all current maintenance for all FMIDs before deploying your product to other environments.

1. Issue the following command to determine which level of RTCS you are running:

   ```
   F RTCS,PACKAGE MAINT KERNEL
   ```

2. The PACKAGE RMID for PROGRAM OBJECT=OSZKERNEL will display a PTF number. If that PTF number does not meet the minimum PTF level required by the products that interact with RTCS, obtain the current maintenance for RTCS.

   **Note**

   - HOLD doc for RTCS PTFs might indicate that an IPL should occur after you apply maintenance. However, doing so is not generally required.
   - In the install dialog, if you deselect RTCS from being downloaded because you already have it installed, the dialog will display the PTFs required to support the products that you are installing.

Controlling sharing of the RTCS system registry

You must have an RTCS started task on each LPAR where a DBC has a member. You can share the RTCS system registry among all or some of the LPARs in a sysplex.
Use the appropriate procedure (for a new RTCS installation or an existing installation) to control whether and how the registry is shared.

**Controlling whether a new RTCS system registry is shared**

If you are installing RTCS for the first time, use this procedure to control sharing of the RTCS system registry.

---

**Note**

The BMC products can operate with a shared or non-shared RTCS system registry. However, in some situations, using a non-shared RTCS registry is better. For more information, see Avoiding ZDBCssid on page 149.

---

1. Review the CHKZMC1 step from the $310VZMC job and the OSZINI\(nn\) member in your JCL data set (where \(nn\) is your SYSCLONE value). The $INCINF member in the JCL data set contains the values used for the symbolics in the job.

   The CHKZMC1 step creates the RTCS system registry; this data set name is referenced on the SREGVLDS parameter in the OSZINI\(nn\) member that RTCS uses at startup.

2. Take the appropriate action based on whether you want to share the RTCS system registry:

   - If you *want to share* the registry across some or all LPARs in a sysplex, ensure that OSZINI\(nn\) contains:
     - An SREGVLDS parameter that references the same system registry data set for those RTCS instances
     - The NOPRIVATE-REGISTRY and SHARED-REGISTRY parameters
     - The same REGISTRY-XCF-GROUP value for those RTCS instances to allow communication

   - If you *do not want to share* the registry, ensure that OSZINI\(nn\) contains:
     - An SREGVLDS parameter that refers to a unique data set
     - The PRIVATE-REGISTRY and NOSHARED-REGISTRY parameters
     - No REGISTRY-XCF-GROUP value

---

**Note**

For more information about RTCS parameters, see the *BMC Runtime Component System Configuration and Administration Guide*.
**Determining whether an existing RTCS system registry is shared**

1. Issue the following console command:
   
   ```
   RO *ALL,F RTCS,REGISTRY STATUS
   ```

2. Look for message OSZ0240I in the output.
   - If the message shows the same data set name from each LPAR, the registry is shared.
   - If the data set names differ, the registry is not shared.

3. *(If generated)* Run steps LGCDFLT and LGCRGRP from the $463LGCD job as follows, based on whether the registry is shared:
   - If the registry is shared, run the steps from any LPAR in the sysplex to register all DBC groups.
   - If the registry is not shared, run the steps on each LPAR where the DBC has a member.

**Checking your cross-system coupling facility (XCF) groups names**

BMC products that use the infrastructure components use the following cross-system coupling facility (XCF) groups:

- The DBC starts an XCF group based on the DBCGROUP parameter on the DBC$STC started task. The DBCGROUP parameter is also used in some configuration jobs with the value specified as DVDBCGID in the $INCINF member in the JCL data set.

- The System and SQL Performance for DB2 products use a DOM agent that runs in the DBC. The DOM agent starts an XCF group based on the DOMPLEX option set name specified in the `<PARMS>` element DOMPLEX= keyword in the $495SDOM job. In that job, the keyword is a symbolic referenced in the $$INCINF member in the JCL data set.

This XCF group should not be shared with other processes. If the DOMPLEX option set name is the same as the DBC GROUP name, the DOM code will add $ to the end of the name (or change the last character to $) to make the name unique. Doing so ensures that the XCF group differs from the one used by the DBC, even if the DBC group name matches the DOMPLEX option set name. The XCF group must be the same for all DOM agents in order to show data collected for remote DB2 subsystems.
The LGC agent starts an XCF group based on what you specify in the $463LGCD job (LGCRGRP step).

The BRD agent starts an XCF group based on what you specify in the BRD option set.

**WARNING**

Use caution when specifying the @ symbol on the XCF group. This symbol can cause translation issues.

All XCF groups must have different names.

**Example**

- Assume that the DBC group and DOMPLEX option set have the same name, DCPLEX. The group names could be as follows:
  - DBC XCF group = DCPLEX
  - DOM XCF group = DCPLEX$ (with DCPLEX automatically changed to DCPLEX$ by the DOM code)
  - LGC XCF group = LGCGRP01
  - BRD XCF group = BRDXCFG
  - NGL XCF group = NGLPLXA

- Assume that the DBC group and DOMPLEX option set have different names. The group names could be as follows:
  - DBC XCF group = DCPLEX
  - DOM XCF group = TESTPLEX
  - LGC XCF group = LGCTEST
  - BRD XCF group = BRDXCFT
  - NGL XCF group = NGLPLXA

**To check XCF group names and enable communication between groups**

1. To display XCF groups that are already in use, issue the following console command:

   ```
   D XCF,GROUP
   ```

2. To determine the names of the XCF groups that are in use, review the DBC$STC started task log and search for XCF.
Note
You can specify a different name for the XCF group used by the DBC. To do so, use the `<XCFGROUP>myxcf</XCFGROUP>` element in the member referenced in the DBCPARMS DD in the DBC$STC started task PROC. The variable `myxcf` represents the XCF group name to be used by the DBC.

3 To enable communication across XCF groups for all components from one LPAR to another, observe the following guidelines:

- Both DBC subsystems should be defined to the same DBC group and should reside in the same sysplex.
- Both DBC subsystems should join the same XCF group that the DBC uses.
- Both DOM agents should join the same XCF group as defined by using the same DOMPLEX option set.
- Both LGC agents should join the same XCF group.
- Both BRD agents should join the same XCF group.
- Both NGL agents should join the same XCF group using the same RG parameter.

Modifying the DBC started task

Running the DBC$STC started task procedure starts the DBC subsystem.

Before you begin

The DBC SSID referenced in the started task procedure must match what you are using for this LPAR. You can use the same started task member on each LPAR by specifying the SSID value in the startup command.

Figure 23 on page 114 illustrates that the DBC SSID is also used for the NGL REGISTRY name, as a method to keep it unique per DBC task, but this is not a requirement.

Figure 23: Example of DBC started task

```
//DBC    PROC  ACC=4111,
// SSID=DC01,       ===> SSID
// G=FDR1PLEX,      ===> GROUP
// T=NO,            ===> TRACE
// TIM=1440         ===> 
//*
//DBC      EXEC PGM=DBCMAIN,REGION=0M,ACCT=&ACC,TIME=&TIM.
```
Preparing for installation

```c
// PARM='SSID=&SSID,GROUP=&G,TRACE=&T'
//STEPLIB DD DSN=BMCPERF.TEST.BMCLINK,
//       DISP=SHR
//DBCPRINT DD SYSOUT=*,RECFM=VA
//SYSPRINT DD SYSOUT=*,RECFM=VA
//BMCPWD DD DISP=SHR,
//         DSN=BMCPERF.TEST.BMCLINK
//SYSTSPRT DD SYSOUT=*
//SYSTERM DD SYSOUT=*,RECFM=VA
//DBCPARMS DD DISP=SHR,
//          DSN=BMCPERF.TEST.UBMCSAMP($DBC&SSID)
//DBCSECUR DD DISP=SHR,
//         DSN=BMCPERF.TEST.UBMCSAMP($SEC&SSID)
//REGISTRY DD DISP=SHR,
//          DSN=BMCPERF.TEST..&SSID.NGLRGY
//DOMTMPLT DD DISP=SHR,
//          DSN=BMCPERF.TEST.BMCTMPLT
//DOMPARMS DD DISP=SHR,
//          DSN=BMCPERF.TEST.BMCPARM
```

**Note**
The DOMTMPLT and DOMPARMS DD statements are used only by System and SQL Performance products. These DD statements might be missing or commented out if you do not have those products.

---

To modify the DBC started task

1. *(optional)* Change the ACC= value at the top of the PROC (if necessary) for accounting information.

2. *(optional)* Add a step in the PROC to APF authorize anything in your STEPLIB, as shown in the following example:

   ```c
   //AUTH1 EXEC PGM=JTHAPF00,ACCT=&ACC,PARM='DBC,STEPLIB'
   ```

To setup another DBC

The DBC$STC started task references two members for the DBC SSID:

- $DBC&ssid
- $SEC&ssid

1. Replicate the members in the appropriate user or runtime data set to use the naming convention for the DBC SSID for this LPAR.

2. Edit the $DBC&ssid member to make the DBC Repository unique for each LPAR, perform one of the following tasks:
   - Change the DBC repository name for this LPAR
Use the `&SSID` symbolic for the repository name in that member. The SSID value is resolved to be the DBC SSID from the SSID parameter in the DBC$STC PROC.

```
<NAME>BMCPERF.&SSID..DBCREPOS</NAME>
```

**Note**
The `$SEC&ssid` member contains only one parameter. If you want this parameter to be the same for all DBC subsystems in this DBC group, hard code the member name and share it among all DBC subsystems.

---

**Determining the default DBC group**

You can have only one default DBC group per RTCS system registry. After the default DBC group is established, changing it can have unexpected results.

Establishing a default DBC Group in the RTCS system registry allows products to obtain their option sets from LGC without having to specify a DBC to connect to. The product will always connect to the default DBC unless a DBC SSID parameter is used.

If you have more than one DBC started task on an LPAR, or have multiple DBCs in the same sysplex using different DBC groups and the RTCS system registry is shared, only one DBC can be registered as the default DBC. If the LGC option sets for any LGC clients are in a different LGC registry from the default DBC, then those DBCs will need to use a special ZDBC.ssid designation. For more information, see “Replicating the DBC by using a different DBC group” on page 142.

If you are not certain which DBC group is the default, use the following procedure to find the default.

**To find the default DBC group**

1. Add the following XML string as input to the LGCUTIL step:

   ```xml
   <lgcutil version="1010">
     <list><dbc /></list>
   </lgcutil>
   ```

   **Note**
   This XML code is also available in the LGC$LIST member in the SAMP data set.

   In the LGCLGPLPR output, the name of the default DBC group (DCPLEX in the following example) is marked with an asterisk.
Working with agents

This section describes some considerations for applying maintenance to agents that run within the DBC subsystem.

The $465INIT and $480INIT jobs define agents to the DBC subsystem. As generated by the installation, these jobs contain <LOADLIB> elements for each agent that starts under the DBC subsystem. Depending on the option you chose for configuring runtime data sets, multiple <LOADLIB> elements per agent might reference the specific data sets that the agent uses.

The DBC started task is meant to be a long-running address space service that remains active for the life of an IPL, similar to the RTCS started task. The purpose of the <LOADLIB> elements is to enable bouncing an agent (not the DBC started task) when maintenance is applied.

As generated by the Installation System, only the load libraries that the agent needs are used in the definition. If you specified to use a single runtime load library, that same library will be in the definition for each agent. The DBC started task needs to be bounced only when DBC maintenance is applied when <LOADLIB> elements are specified for the agents, even if the task contains load library members used by other agents.

The DBC creates a TASKLIB consisting of any libraries in an agent's <LOADLIB> elements, followed by any libraries in the STEPLIB. If neither of those have data sets defined to them, the load modules should be in LINKLIST. The product searches in the following order:

- <LOADLIB> elements, if present
- STEPLIB data sets, if present
- LINKLIST

To display the data sets being used by the agents in the DBC

1. Issue the following console command on the LPAR where the DBC started task is up:
   
   
   ```
   /ssid DBC,DISPDSNS
   ```

   where `ssid` is the DBC subsystem identifier.

2. Review the output in the DBCPRINT DD of the DBC$STC started task

Chapter 5  Deploying products that use the infrastructure components  117
In the example below, the data sets with (PRODUCT) are defined in the <LOADLIB> tags of the Product. The data sets with (STEPLIB) are defined in the STEPLIB of the DBC$STC started task.

### Example

<table>
<thead>
<tr>
<th>CTAJ</th>
<th>Products With Allocated DSNs</th>
<th>-------</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTAJ</td>
<td>Product DBC.ZDBC000</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(STEPLIB) SCC.INST1110.BMCLINK</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>Product BRD.ZBRD000</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(PRODUCT) DOM.ZDOM000</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(PRODUCT) BB.XTST9.BBLINK</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(PRODUCT) BB.XTST9.XXLINK</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(PRODUCT) AFDQA.ALL.PASSWORD</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(STEPLIB) SCC.INST1110.BMCLINK</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>Product GUD.ZGUD100</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(PRODUCT) SCC.INST1110.BMCLINK</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(PRODUCT) PRF.INST1110.BMCLINK</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(PRODUCT) ADM.INST1110.BMCLINK</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>Product LGCA10</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(STEPLIB) SCC.INST1110.BMCLINK</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>Product NGL.ZNGL000.MVQI</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(PRODUCT) BB.XTST6.XXLINK</td>
<td></td>
</tr>
<tr>
<td>CTAJ</td>
<td>(STEPLIB) SCC.INST1110.BMCLINK</td>
<td></td>
</tr>
</tbody>
</table>

You can change the load library information in the <LOADLIB> elements by using one of the following methods:

- “Removing references to the load library” on page 123
- “Changing references to the load library” on page 118

You can deploy maintenance to the agents while the DBC address space continues to run. For details, see “Deploying maintenance to agents” on page 124.

### Changing references to the load library

Use the following procedure if you want to keep the <LOADLIB> elements but reference a different data set or change the LOADLIB to reference an alias.

**Note**

You can use this procedure even if you have already run the $465INIT or $480INIT job.

1. Edit the $465INIT and $480INIT installation jobs, depending on which agent's <LOADLIB> elements you want to change.

2. Change the data set names in the <LOADLIB> elements in each agent definition.

As generated by the installation system, symbolics are referenced for the data set names. As an alternative, you can change these to have fully defined data set names.
To change the symbolic values, copy the SET statements in the $INCINF member in your JCL data set to the $INCUSR member and update the values. Any symbolic values defined in the $INCUSR member will override the values in the other $INC members.

The installation system provided three options for naming Low Level Qualifiers (LLQ) for your runtime data sets:

1. Use BMC* LLQs (recommended)
   Use BMCLINK as the runtime LLQ in the <LOADLIB> elements.

2. Use BB/DB/IM/XX LLQs
   Use column 3 in Table 16 on page 119 for the <LOADLIB> data set LLQs.

3. Customize the LLQs
   Use the SMP/E target load library names in column 2 in Table 16 on page 119 to map to the LLQs that you chose for those libraries. The OZInnnn member in the JCL data set (where nnnn is a timestamp) contains the mapping.

Table 16 on page 119 lists the SMP/E target load library names and the corresponding runtime LLQs if option 2 for runtime LLQs was chosen.

**Table 16: Load library names required by agents**

<table>
<thead>
<tr>
<th>Step name</th>
<th>SMP/E Target load library names</th>
<th>For runtime LLQ option #2: Use BB/DB/IM/XX LLQs</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRDINIT</td>
<td>Does not require any load library elements</td>
<td></td>
</tr>
<tr>
<td>GUDINIT</td>
<td>PSSLINK</td>
<td>BBLINK</td>
</tr>
<tr>
<td></td>
<td>ACSLINK</td>
<td>DBLINK</td>
</tr>
<tr>
<td></td>
<td>ACTLINK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AEXLINK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCCLINK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data set that contains ACTDOPTS</td>
<td></td>
</tr>
<tr>
<td>DIGBLB</td>
<td></td>
<td>XXLINK</td>
</tr>
<tr>
<td>DIGB4LB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUDLINK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LGCLINK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MRELINK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USCLINK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Preventing Privileged Access to System Resources

Chapter 5  Deploying products that use the infrastructure components  119
<table>
<thead>
<tr>
<th>Step name</th>
<th>SMP/E Target load library names</th>
<th>For runtime LLQ option #2: Use BB/DB/IM/XX LLQs</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOMINIT</td>
<td>DASLINK                                           DOMLINK                                          PSSLINK</td>
<td>BBLINK</td>
</tr>
<tr>
<td></td>
<td>Also, the following names apply if the associated products are installed:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BDSLINK (MainView for DB2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IODLINK (APPTUNE for DB2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PMDLINK (Pool Advisor for DB2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PSELINK (SQL Explorer for DB2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPDLINK (System Performance for DB2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCCLINK</td>
<td>DBLINK</td>
</tr>
<tr>
<td></td>
<td>DIGBLB                                            DIGB4LB                                          LGCLINK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MRELINK                                           NGLLINK                                          USCLINK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>XXLINK</td>
</tr>
<tr>
<td>LGCINIT</td>
<td>SCCLINK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DIGBLB                                            DIGB4LB                                          LGCLINK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MRELINK                                           NGLLINK                                          USCLINK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>XXLINK</td>
</tr>
<tr>
<td>NGLINIT</td>
<td>SCCLINK</td>
<td></td>
</tr>
<tr>
<td>NGLAGJV</td>
<td>NGLLINK</td>
<td></td>
</tr>
<tr>
<td>MVINIT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Step name | SMP/E Target load library names | For runtime LLQ option #2: Use BB/DB/IM/XX LLQs
--- | --- | ---
a | The GUD agents used by BMC Workbench require that the DB2 DSNLOAD library be included in the GUD concatenation. Add the DB2 DSNLOAD library to the <LOADLIB> tags in the GUDINIT step in the $480INIT job. **Note:** If the DB2 DSNLOAD library is included in the LINKLIST, then you do not need to include it in the concatenation. | b | If you have selected runtime LLQ option #1, all the data set names will be BMCLINK.

---

**Table 17 on page 121** lists load libraries that the DBC needs.

**Table 17: Load library names required by the DBC**

<table>
<thead>
<tr>
<th>SMP/E Target load library names</th>
<th>For runtime LLQ option #2: Use BB/DB/IM/XX LLQs</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCCLINK</td>
<td>DBLINK</td>
<td></td>
</tr>
<tr>
<td>DBCLINK</td>
<td>XXLINK</td>
<td></td>
</tr>
</tbody>
</table>

**Note**
The referenced load library data sets listed in **Table 16 on page 119** and **Table 17 on page 121** must be APF authorized.

3 **(APPTUNE only)** The REGIOD step of the $490RGIM job references the load library containing the DOMQLGCV module. If required, change the load library name in the <files> tag of the job and rerun the REGIOD step. The change will take effect when the DOM agent is recycled.

4 Run the jobs.

5 Issue the REFRESH or START/STOP commands for each agent that is affected by the change.
Note
(only NGL) You must stop any NGL client products that require NGL before stopping their NGL agents. For example, DOM and MVL are NGL client products that run as DBC agents. Following is an example of stopping the DOM agent in the DBC, refreshing its NGL piid DOM1, and then starting the DOM DBC agent again.

/ssid DOM,STOP
/ssid NGL,REFRESH,DOM1
/ssid DOM,START

Other NGL client products such as MainView Transaction Analyzer and MainView for MQ do not have agents that run in the DBC, but rather have address spaces that need to be stopped. Review the product-specific configuration guides for the specific commands to stop these kind of NGL clients.

WARNING
Not stopping the NGL client products that require NGL before recycling their NGL agents, can result in the clients and agents running at different levels which causes problems.

If you are changing the NGR (NREG) agent, then all NGL agents should be stopped first which causes the NGR agent to be stopped automatically. Then issue the START for each of the NGL agents. The NGR agent will automatically start when the first NGL agent starts. In the example below, DOM and MVL are DBC agents that use NGL agents DOM1 and MVI1 respectively. NGL agent MVT1 is also active but does not use a DBC agent:

/ssid DOM,STOP
/ssid MVL,STOP
/ssid NGL,STOP,DOM1
/ssid NGL,STOP,MVI1
/ssid NGL,STOP,MVT1
Etc for all the rest of the NGL piids
/ssid DBC,DISPAGENTS  - ensure that no NGL or NGR agents are active.
/ssid NGL,START,DOM1
/ssid NGL,START,MVI1
/ssid NGL,START,MVT1
Etc for all the rest of the NGL piids
/ssid DOM,START
/ssid MVL,START

Table 18 on page 122 lists the command to issue for each component. Issue the commands in the order shown. The ssid value represents the identifier of the DBC subsystem.

Table 18: Commands to refresh agents

<table>
<thead>
<tr>
<th>To refresh agent</th>
<th>Issue this console command</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRD</td>
<td>/ssid BRD,REFRESH</td>
</tr>
<tr>
<td>LGC</td>
<td>/ssid LGC,REFRESH</td>
</tr>
<tr>
<td>To refresh agent</td>
<td>Issue this console command</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------</td>
</tr>
</tbody>
</table>
| NGL c           | `/ssid NGL,REFRESH, piid`  
The `piid` value identifies the NGL ID. 
To stop and then start, use:  
`/ssid NGL,STOP,piid`  
`/ssid NGL,START,piid` |
| GUD b           | `/ssid GUD,STOPALL`  
followed by  
`/ssid GUD,STARTALL` |
| DOM a           | `/ssid DOM,REFRESH` |
| MVL             | `/ssid MVL,STOP`  
`/ssid MVL,START` |

a The NGL and LGC agents should be active before issuing this command.
b The LGC agent should be active before issuing this command.
c If installed, the DOM and MVL agents should be stopped before issuing this command, and then started after the NGL agents are active again.

### Removing references to the load library

The `$465INIT` and `$480INIT` jobs reference the product LOADLIB in a few places. Use the following procedure if you want to remove references to the LOADLIB data sets. The load modules must then be accessible from the DBC STEPLIB or LINKLIST.

**Note**

You can use this procedure even if you have already run the `$465INIT` or `$480INIT` job.

1. Edit the `$465INIT` and `$480INIT` installation jobs.
2. Delete all `<LOADLIB>` lines in the jobs.

3. *(APPTUNE only)* The REGIOD step of the `$490RGIM` job references the load library containing the DOMQLGCV module. If desired, delete the load library name in the `<files>` tag of the job and rerun the REGIOD step. The change will take effect when the DOM agent is recycled.

4. Run the jobs.

5. Issue the REFRESH or START/STOP commands for each agent that is affected by the change.
For information about the REFRESH commands, see Table 18 on page 122.

**Note**

*(only NGL)* You must stop any NGL client products that require NGL before stopping their NGL agents. For example, DOM and MVL are NGL client products that run as DBC agents. Following is an example of stopping the DOM agent in the DBC, refreshing its NGL *piid* DOM1, and then starting the DOM DBC agent again.

```
/ssid DOM,STOP
/ssid NGL,REFRESH,DOM1
/ssid DOM,START
```

Other NGL client products such as MainView Transaction Analyzer and MainView for MQ do not have agents that run in the DBC, but rather have address spaces that need to be stopped. Review the product-specific configuration guides for the specific commands to stop these kind of NGL clients.

**WARNING**

Not stopping the NGL client products that require NGL before recycling their NGL agents, can result in the clients and agents running at different levels which causes problems.

## Deploying maintenance to agents

This task describes how to apply maintenance to the agents that run within the DBC.

### To apply maintenance for agents

1. Issue the following console command on the LPAR where the DBC started task is up: `/ssid DBC,DISPDSNS` where `ssid` is the DBC subsystem identifier.

2. Review the output in the DBCPRINT DD in the active DBC started task.

   In the DISPDSNS display, any data sets labeled as *PRODUCT* come from `<LOADLIB>` elements associated with that product.

3. Review the PTFs that are being applied to determine which FMIDs will be affected.

4. Based on how you organized your runtime load libraries in the installation, cross-reference these lists with Table 16 on page 119 to determine which agents will need to be refreshed.

5. Issue the STOP command for each agent that is receiving new maintenance.
BMC recommends deploying new maintenance when no users will be attempting to use the product. Stopping agents when users are in the product can cause error messages that indicate the agent is stopping or is unavailable.

Table 19 on page 125 lists commands to stop agents. Issue these commands in the order shown. The ssid value represents the identifier of the DBC subsystem.

Table 19: Commands to stop agents

<table>
<thead>
<tr>
<th>To stop agent</th>
<th>Issue this console command</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRD</td>
<td>/ssid BRD,STOP</td>
</tr>
<tr>
<td>DOM</td>
<td>/ssid DOM,STOP</td>
</tr>
<tr>
<td>GUD</td>
<td>/ssid GUD,STOPALL</td>
</tr>
<tr>
<td>LGC</td>
<td>/ssid LGC,STOP</td>
</tr>
<tr>
<td>To stop agent</td>
<td>Issue this console command</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>NGL</td>
<td><code>/ssid NGL,STOP, piid</code></td>
</tr>
<tr>
<td></td>
<td>The <code>piid</code> value identifies the NGL ID.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> <em>(NGL only)</em> You must stop any NGL client products that require NGL before stopping their NGL agents. For example, DOM and MVL are NGL client products that run as DBC agents. Following is an example of stopping the DOM agent in the DBC, refreshing its NGL <code>piid</code> DOM1, and then starting the DOM DBC agent again.</td>
</tr>
<tr>
<td></td>
<td><code>/ssid DOM,STOP</code></td>
</tr>
<tr>
<td></td>
<td><code>/ssid NGL,REFRESH,DOM1</code></td>
</tr>
<tr>
<td></td>
<td><code>/ssid DOM,START</code></td>
</tr>
<tr>
<td></td>
<td>Other NGL client products such as MainView Transaction Analyzer and MainView for MQ do not have agents that run in the DBC, but rather have address spaces that need to be stopped. Review the product-specific configuration guides for the specific commands to stop these kind of NGL clients.</td>
</tr>
<tr>
<td></td>
<td><strong>WARNING:</strong> Not stopping the NGL client products that require NGL before recycling their NGL agents can result in the clients and agents running at different levels which causes problems.</td>
</tr>
<tr>
<td></td>
<td>If you are changing the NGR (NREG) agent, then all NGL agents should be stopped first which will cause the NGR agent to be stopped automatically. Then issue the START for each of the NGL agents. The NGR agent will automatically start when the first NGL agent starts. In the example below, DOM and MVL are DBC agents that use NGL agents DOM1 and MVI1 respectively. NGL agent MVT1 is also active but does not use a DBC agent:</td>
</tr>
<tr>
<td></td>
<td><code>/ssid DOM,STOP</code></td>
</tr>
<tr>
<td></td>
<td><code>/ssid MVL,STOP</code></td>
</tr>
<tr>
<td></td>
<td><code>/ssid NGL,STOP,DOM1</code></td>
</tr>
<tr>
<td></td>
<td><code>/ssid NGL,STOP,MVI1</code></td>
</tr>
<tr>
<td></td>
<td><code>/ssid NGL,STOP,MVT1</code></td>
</tr>
<tr>
<td></td>
<td>Etc for all the rest of the NGL <code>piids</code></td>
</tr>
<tr>
<td></td>
<td><code>/ssid DBC,DISPAGENTS</code>    - ensure that no NGL or NGR agents are active</td>
</tr>
<tr>
<td></td>
<td><code>/ssid NGL,START,DOM1</code></td>
</tr>
<tr>
<td></td>
<td><code>/ssid NGL,START,MVI1</code></td>
</tr>
<tr>
<td></td>
<td><code>/ssid NGL,START,MVT1</code></td>
</tr>
<tr>
<td></td>
<td>Etc for all the rest of the NGL <code>piids</code></td>
</tr>
<tr>
<td></td>
<td><code>/ssid DOM,START</code></td>
</tr>
<tr>
<td></td>
<td><code>/ssid MVL,START</code></td>
</tr>
<tr>
<td>MVL</td>
<td></td>
</tr>
</tbody>
</table>

6 Copy the changed members from your target zone to your runtime libraries, observing the following guidelines:

**Note**

Maintenance usually updates the load library but can sometimes include other libraries such as PLIB, XML, and so on. Do not forget to copy these libraries to the runtime data sets, too.
If you have an alias defined to your runtime load library, and you typically copy new maintenance to a second load library, then delete the alias from the old library and define it to the new library.

If you apply maintenance to a new set of libraries and want to use them in place of the current runtime data sets, follow the instructions in “Changing references to the load library” on page 118.

If you are using LINKLIST instead of <LOADLIB> elements or the DBC STEPLIB, issue the `LLA REFRESH` command.

7 Issue the START command for each agent that is receiving new maintenance and that was previously stopped.

Table 20 on page 127 lists commands to start agents. Issue the commands in the order shown. The `ssid` value represents the identifier of the DBC subsystem.

<table>
<thead>
<tr>
<th>To start agent</th>
<th>Issue this console command</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRD</td>
<td><code>/ssid BRD,START</code></td>
</tr>
<tr>
<td>DOM</td>
<td><code>/ssid DOM,START</code></td>
</tr>
<tr>
<td>Note: The NGL and LGC components should be active before you issue the START command to the DOM agent.</td>
<td></td>
</tr>
<tr>
<td>GUD</td>
<td><code>/ssid GUD,STARTALL</code></td>
</tr>
<tr>
<td>LGC</td>
<td><code>/ssid LGC,START</code></td>
</tr>
<tr>
<td>NGL</td>
<td><code>/ssid NGL,START, piid</code></td>
</tr>
<tr>
<td>The <code>piid</code> value identifies the NGL ID.</td>
<td></td>
</tr>
<tr>
<td>MVL</td>
<td></td>
</tr>
</tbody>
</table>

To apply maintenance to the DBC

1 Ensure that all users have been notified of the maintenance window.

2 Stop the DBC started task by using the following console command, where `dbcid` is the DBC subsystem ID:
   ```
   p ssid
   ```

3 Copy the changed members from your target zone to your runtime libraries.
Note
Maintenance usually updates the load library but can sometimes also include other libraries such as PLIB, XML, and so on. Do not forget to copy these libraries to the runtime data sets too.

■ If you have an alias defined to your runtime load library, and you typically copy new maintenance to a second load library, you must delete the alias from the old library and define it to the new library.

■ If you apply maintenance to a new set of libraries and want to use them in place of the current runtime data sets, edit the DBC$STC PROC and change the STEPLIB to point to the new runtime data sets.

■ If you are using LINKLIST instead of <LOADLIB> elements issue the LLA REFRESH command.

4 Start the DBC started task by issuing the following console command, where dbcproc is the name of the DBC$STC started task member in your PROCLIB:

```
s dbcproc
```
If you already have any of the infrastructure components, you should choose the option in the Installation System to reuse these components. The Installation System guides you through recycling the DBC started task if DBC maintenance has been applied; otherwise, only the agents running in the DBC will be recycled. The installation jobs will be generated to reference the new load libraries, update any product definitions, and then stop and restart the agents with the new code. Any new product agents will be defined to the DBC and started. Some products do not have agents but use existing component agents (that is, LGC and NGL). Products that use option sets are registered to the LGC private registry with their new template definitions.

**Note**
*(only NGL)* You must stop any NGL client products that require NGL before stopping their NGL agents. For example, DOM and MVL are NGL client products that run as DBC agents. Following is an example of stopping the DOM agent in the DBC, refreshing its NGL *piid* DOM1, and then starting the DOM DBC agent again.

```
/ssid DOM,STOP
/ssid NGL,REFRESH,DOM1
/ssid DOM,START
```

Other NGL client products such as MainView Transaction Analyzer and MainView *for MQ* do not have agents that run in the DBC, but rather have address spaces that need to be stopped. Review the product-specific configuration guides for the specific commands to stop these kind of NGL clients.

**WARNING**
Not stopping the NGL client products that require NGL before recycling their NGL agents, can result in the clients and agents running at different levels which causes problems.

### Reusing RTCS

In some environments, RTCS is managed by a different group in a different SMP/E zone. In this scenario, you should choose the option in the Installation System to deselect RTCS. Doing so prevents it from being downloaded in your new zone and from having more than one copy of the code, which could cause confusion over maintenance levels.

**Note**
When deselecting RTCS, a panel in the Installation System will indicate the PTF level required by the products that will be interacting with RTCS. Ensure that you apply this level of maintenance before using those products.

If you decide to download RTCS into your SMP/E zone, you will have the opportunity to reuse your existing RTCS. An LPAR can have only one active RTCS started task. If you already have RTCS installed and you plan to apply all the maintenance from the new zone, indicate that you want to reuse RTCS in the
Installation System. Doing so prevents you from having to answer configuration questions in the Installation System for RTCS. To have RTCS use the new code, perform the following steps:

1. Replace or update the existing RTCS load libraries using the existing library names.

2. If you are in a continuous operation mode, review each PTF, and proceed accordingly:
   - If none require an RTCS Cold Start, refresh the packages using the RTCS REFRESH command.
   - If an RTCS Subsystem Restart is desired:
     1. Bring down the current RTCS started task.
     2. Bring up the new RTCS started task referencing the updated libraries.

3. Restart the RTCS started task.

   **Note**
   The products that interact with RTCS require a minimum maintenance level. If the minimum level is already present, you can delay recycling RTCS until a more convenient time. If the products do not detect the required minimum PTF level, they display an error message.

If you are installing RTCS for the first time, do not deselect it from being downloaded, and do not indicate that you want to reuse it. The Installation System will help you configure RTCS and provide instructions about how to start RTCS.

**Reusing DBC**

If you already have a DBC started task, you also have a DBC repository that contains product and agent definitions and commands. You should reuse this repository from release to release to prevent any disruption from upgrading products or adding new products. If you have an existing DBC, you should indicate that you want to reuse the DBC in the Installation System. Doing so prevents you from having to answer configuration questions in the Installation System for DBC.

The Installation System downloads the DBC component files along with all current maintenance. After you apply DBC maintenance, you must recycle the DBC started task to load the new code. You should compare your current DBC started task and parameters with the newly generated task to determine whether the products that you are installing require any changes.
Reusing NGL

If you already have an NGL agent running in the DBC, you have an NGL registry data set that contains information about each of its PIIDs and LOGSETS. To avoid losing that information and having to reconfigure the component, indicate that you want to reuse the NGL in the Installation System. Even if the NGL has a new release number and FMID, you can upgrade the NGL in place.

The Installation System downloads the NGL component files along with all current maintenance. The generated installation jobs will reference the new libraries and stop and restart the NGL agents.

Reusing LGC

If you already have an LGC agent running in the DBC, you have an LGC datastore that contains option sets for other BMC products. To avoid losing that information, you should indicate that you want to reuse the LGC in the Installation System. The Installation System downloads the LGC component files along with all current maintenance. The generated installation jobs will reference the new libraries, register the new LGC templates, and then stop and restart the LGC agent.

Adding new products to the DBC

All product definitions will be added or updated in the DBC repository in either of the following cases:

- You have an existing DBC with BMC products defined to it.
- You are installing the DBC for the first time.

Installing and upgrading the DOM agent for System and SQL Performance products

This procedure applies only to System and SQL Performance products which began using the infrastructure components in version 10.1.00. The installation process varies depending on whether you are migrating from a previous version.

- For System and SQL Performance products, if you are installing the DOM agent for the first time (not migrating from a previous release), follow all the installation instructions and run the jobs in order.
If you are migrating from version 10.1.00 or later of the System and SQL Performance products, which use the DOM agent, take the following into consideration:

Upgrading from version 10.1.00 requires the `<IGNORE>` element in the DBC startup parameters. In later versions, the product is defined with ZDOM000 to the DBC so that it can be upgraded in place and the `<IGNORE>` element is no longer be necessary.

The System and SQL Performance products define a DOM agent (that is, the Data Collector) that runs in the DBC address space. Only one version of the DOM agent can be active in the DBC. When migrating the DOM agent from version 10.1.00, the Installation System will generate a DBC$ssid member that contains an `<IGNORE>` element for the version 10.1 of the DOM agent. Doing so prevents that version from starting and allows the new version to become active.

**Note**

Ensure that the DBC$STC job references the generated DBC$ssid member that contains the new syntax.

---

**Replicating the installation to other DB2 subsystems**

The procedures in this section offer two methods for replicating DB2 objects to other DB2 subsystems. You can use the Multiple Subsystem Install (MSSID) method through the Installation System, or you can use product deployment.

**Note**

BMC products use product-specific DB2 objects for some features. The same product code base can be used with many DB2 subsystems at differing DB2 versions and modes. The procedures in this section are based on the following assumptions:

- Only the DB2 objects need to be created.
- The product suite of data sets from another installation will be used to run the relevant jobs.

Use the procedure that best suits your situation:

- The Installation System's MSSID method replicates the DB2 objects to other DB2 subsystems with just a few changes to the original DB2 subsystem values. This method generates a JCL data set for each subsequent subsystem.

- Product deployment requires running a job to copy the JCL data set and then changing some values in the INCLUDE members before running the jobs.
To replicate DB2 objects by using MSSID

1. Invoke the installation CLIST.

2. Select the install project that was used for the full install on the initial DB2 subsystem.

3. From the Execute Project panel, select Configuration Option 6 Multiple SSID install for DB2 Products only.

4. Complete the MSSID options for the DB2 subsystems for which you want to replicate the installation.

   You can change the job card and other options for each subsystem, if necessary.

   When you finish specifying options, the Installation System generates the $S00JCL job.

5. Run the generated $S00JCL job.

   This job replicates a JCL data set for each DB2 subsystem. Each data set contains jobs to run for that subsystem, customized with the correct DB2 libraries.

6. Run the $700 series installation jobs for each DB2 subsystem, taking the following details into consideration

   - Before running the $729DOP1 job (if generated):

     — If your original install specified to migrate the DOPTS modules for your products, the job will include a `pppDOPT` step (where `ppp` is the product code) to copy the existing DOPTS modules forward to the new version. The job will also include a `pppMERGE` step to create a new DOPTS module from default values, but this step will be commented out.

     — If your new environment does not have any DOPTS modules to carry forward, then comment out the `pppDOPT` step and uncomment the `pppMERGE` step. Review the symbolic variables for these steps in the `$INCD$$INCDB2` member. Copy any variables that need to change to the `$INCUSR` member and change the values there.

   - If your original install specified to migrate DB2 data, other migrate jobs will be generated such as `$763MIGP`, `$765MIG$, `$766MIG$, `$767COPY`. Do not run these jobs if your new environment does not have any DB2 data to migrate.

   - For those BMC products that use an LGC option set and generate a `$770IVP` job, ensure that the option set that the job is using exists before running the job. The DBC started task must be running and the LGC agent must be activated in order to access the option set.
To use product deployment

1. Run the $$JCLCPY in the install JCL to make a copy of the original JCL to be used for deployment.

2. Edit the $$INC members to change any values needed for this deployment.

   **Tip**
   Instead of editing each $$INC member for new values, copy the SET statements for variables you want to change to the $$INCUSR member. Any values in the $$INCUSR member will override the values in the other $$INC members. When specifying only the variables that have to change from one system to another in the $$INCUSR member, it speeds up locating and changing the values and helps to keep you organized.

3. Use your own process to transport the runtime data sets to the other environment if necessary.

4. Run all of the $700 series jobs in order.

   **Note**
   BMC recommends that you use the product deployment method if you need to deploy the products to other sysplexes or LPARs. See the Installation System documentation for more information.

---

**Product deployment for infrastructure components**

Product deployment provides a quick way to deploy the BMC products to other environments such as another SYSPLEX, LPAR in the same SYSPLEX, or DB2 subsystem.

The generated JCL uses $$INC members which make it easy to change values that might be specific to certain environments. Job names and job categories make it easy to identify which jobs are needed depending on your scenario.
Consider the following when deploying products that use infrastructure components.

Figure 24: Product deployment scenarios involving infrastructure

Assume that the following conditions apply to this scenario:

- The sandbox area has two LPARs (S1 and S2) in sysplex X1, and you want these two LPARs to share the same runtime data sets.

- In the development environment, sysplexes X1 and X2 share DASD. You want development LPARs D1 and D2 to use different runtime data sets than the sandbox so that you can control when maintenance gets applied.

- The test and production environments have two LPARs each (T1 and T2 for test, and P1 and P2 for production) in sysplex X3. Because X3 does not share DASD with the sandbox, the runtime data sets will need to be transported to X3. The LPARs in X3 share DASD with each other (that is, T1, T2, P1, and P2 share DASD), but you want to use different runtime data sets between test and production so that you can control when maintenance is applied. Data sets for test and production will need to be transported separately and given different HLQs to keep them distinct.

On sysplexes X1 and X2, each LPAR uses its own RTCS system registry. On sysplex X3, the RTCS system registry is shared across all the LPARs.

For this environment, the values for the runtime customization instances could be as shown in Table 21 on page 136:
Table 21: Example of runtime customization values

<table>
<thead>
<tr>
<th>Sysplex</th>
<th>LPAR</th>
<th>Runtime data set HLQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>S1</td>
<td>HLQSAND</td>
</tr>
<tr>
<td>X1</td>
<td>S2</td>
<td>HLQSAND</td>
</tr>
<tr>
<td>X2</td>
<td>D1</td>
<td>HLQDEV</td>
</tr>
<tr>
<td>X2</td>
<td>D2</td>
<td>HLQDEV</td>
</tr>
<tr>
<td>X3</td>
<td>T1</td>
<td>HLQTEST</td>
</tr>
<tr>
<td>X3</td>
<td>T2</td>
<td>HLQTEST</td>
</tr>
<tr>
<td>X3</td>
<td>P1</td>
<td>HLQPROD</td>
</tr>
<tr>
<td>X3</td>
<td>P2</td>
<td>HLQPROD</td>
</tr>
</tbody>
</table>

For the environment in the example shown in this section, the infrastructure data sets should be set up as shown in Table 22 on page 136.

Table 22: Infrastructure components

<table>
<thead>
<tr>
<th>Component</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S1</td>
<td>S2</td>
<td></td>
</tr>
<tr>
<td>RTCS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LGC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NGL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The RTCS system registry and LGC private registry will be shared per set up for the sandbox, development, test, and production environments. In other words, for each separate set of data sets, the RTCS and LGC components will be shared. The DBC and NGL components will have an instance on each LPAR. The DBC will have a unique repository data set per LPAR. NGL can be configured to share the NGL registry with other LPARs. If the NGL registry is shared, BMC suggests that the NGL registry be shared with a like environment. For example, a sandbox with a sandbox, a test with a test, and so on. The NGL registry will be shared per sandbox, development, test, or production environment. The NGL log files, however, can only be active on one LPAR concurrently.
Example of infrastructure component values

Table 23 on page 137 shows an example of names of components and members of the infrastructure components. The example shows unique DBC SSIDs, DBC repositories, NGL registries, NGL IDs, shared LGC repositories, and LGC XCF Groups.

<table>
<thead>
<tr>
<th>DBC SSID</th>
<th>DBC group</th>
<th>DBC repository</th>
<th>NGL registry</th>
<th>NGL XCF group</th>
<th>LGC repository</th>
<th>LGC XCF group</th>
</tr>
</thead>
<tbody>
<tr>
<td>SND1</td>
<td>SANDPLEX</td>
<td>HLQ.SND1.DBCREPOS</td>
<td>HLQ.SND1.NGLRGRY</td>
<td>NGLPLEXS</td>
<td>HLQ.SND1.LGCRGRY</td>
<td>LGCSAND</td>
</tr>
<tr>
<td>SND2</td>
<td>SANDPLEX</td>
<td>HLQ.SND2.DBCREPOS</td>
<td>HLQ.SND1.NGLRGRY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEV1</td>
<td>DEVPLEX</td>
<td>HLQ.DEV1.DBCREPOS</td>
<td>HLQ.DEV.NGLRGRY</td>
<td>NGLPLEXD</td>
<td>HLQ.DEV.LGCRGRY</td>
<td>LGCEDEV</td>
</tr>
<tr>
<td>DEV2</td>
<td>DEVPLEX</td>
<td>HLQ.DEV2.DBCREPOS</td>
<td>HLQ.DEV.NGLRGRY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TST1</td>
<td>TESTPLEX</td>
<td>HLQ.TST1.DBCREPOS</td>
<td>HLQ.TST.NGLRGRY</td>
<td>NGLPLEXT</td>
<td>HLQ.TST.LGCRGRY</td>
<td>LGCTEST</td>
</tr>
<tr>
<td>TST2</td>
<td>TESTPLEX</td>
<td>HLQ.TST2.DBCREPOS</td>
<td>HLQ.TST.NGLRGRY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRD1</td>
<td>PRODPLEX</td>
<td>HLQ.PRD1.DBCREPOS</td>
<td>HLQ.PRD.NGLRGRY</td>
<td>NGLPLEXP</td>
<td>HLQ.PRD.LGCRGRY</td>
<td>LGCPRD</td>
</tr>
<tr>
<td>PRD2</td>
<td>PRODPLEX</td>
<td>HLQ.PRD2.DBCREPOS</td>
<td>HLQ.PRD.NGLRGRY</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note

- The DBC group name is the same per sandbox, development, test, and production environment, respectively. This organization allows communication between the DBC subsystems on LPARS in the sandbox, development, test, and production environment, but not across DBC groups.

- The LGC registry and LGC XCF group are the same per sandbox, development, test, and production environment, respectively. This organization allows sharing option sets in each of those environments, but not between environments. For example, option sets for production will not be shared with the test, development, or sandbox environments.

- The NGL Registry data set includes the environment as one of the nodes since it can be shared. However, each product using NGL should have its own piid (product instance identifier). The piids are referenced in the $480INIT and $948xxxx jobs for the products that use them.

BMC recommends not using a version-specific name for registries and repositories so that they can persist and be easily reused from release to release.
When to change infrastructure values

To use product deployment with the infrastructure components, you must change the values of symbolic variables in the $$INCINF member of your JCL installation data set.

Tip
Instead of editing each $$INC member for new values, copy the SET statements for variables you want to change to the $$INCUSR member. Any values in the $$INCUSR member will override the values in the other $$INC members. When specifying only the variables that have to change from one system to another in the $$INCUSR member, it speeds up locating and changing the values and helps to keep you organized.

The following table lists the symbolic variables contained in the $$INCINF member. Using the deployment scenarios in “Product deployment for infrastructure components” on page 134 as a guide, you will change the values of the infrastructure symbolic variables when your target environment includes a change such as a different DBC group name.

Table 24: Symbolic variables contained in $$INCINF member

<table>
<thead>
<tr>
<th>Product code</th>
<th>Symbolic variable</th>
<th>Variable description</th>
<th>Need different value when deploying to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOM</td>
<td>DVDBCDPL</td>
<td>DOMPLEX NAME</td>
<td>Different DBC Group</td>
</tr>
<tr>
<td>DOMHELP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOMPROF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOMSECU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOMRDIR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product code</td>
<td>Symbolic variable</td>
<td>Variable description</td>
<td>Need different value when deploying to...</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>----------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>NGL</td>
<td>DVNGLREG</td>
<td>NGL Repository</td>
<td>Different DBC started task</td>
</tr>
</tbody>
</table>
|              | DVNGLARC          | NGL Archive PROC name| Different DOMPLEX NAME on shared DASD that uses the same PROCLIB data set for NGLARCH started task  
The PROCLIB and the PROCNAME can be the same across LPARs. |
|              | DVNGLDPI          | NGL PIID for DOM     | Different DBC started task               |
|              | DVNGLMPI          | NGL PIID for MVT     | The NGL agent PIID does not have to be unique for each DBC. If there are multiple NGL MVT agents in the same DBC, each PIID must be unique. |
|              | DVNGLQPI          | NGL PIID for MVM     | The NGL agent PIID does not have to be unique for each DBC. If there are multiple NGL MVM agents in the same DBC, each PIID must be unique. |
|              | DVNGLIPI          | NGL PIID for MV Infra | The NGL agent PIID does not have to be unique for each LPAR. If there are multiple NGL MVI agents running on an LPAR, each PIID must be unique. |
|              | DVNGLJPI          | NGL PIID for MV MJE  | The NGL agent PIID does not have to be unique for each DBC. If there are multiple NGL MVMJE agents in the same DBC, each PIID must be unique. |
|              | Customer provided value | NGL LOGSET for MVT | Each name must be unique |
|              | Customer provided value | NGL LOGSET for MVM | Each name must be unique |
|              | DVMJELLG          | NGL LOGSET for MVMJE | Each name must be unique |
|              | DVMJELLS          | NGL LOGSET prefix for MVMJE | Each name must be unique |
|              | Customer provided value | NGL LOGSET prefix for MVM | Each name must be unique |
|              | Customer provided value | NGL LOGSET prefix for MVT | Each name must be unique |
|              | DVMJELVL          | NGL Volume for MV MJE| (Change as desired) |
Replicating DBCs on shared DASD in the same sysplex

The procedures in this section offer two methods for replicating the DBC in a shared-DASD environment in the same sysplex as the original installation. You can replicate the DBC in the same DBC group as the original installation, or in a different DBC group.

Replicating the DBC by using the same DBC group and option sets

Use this procedure if you have one DBC installed and running, and you want to deploy the installation to another DBC that will join the same DBC group and use the same option sets.

Note

Ensure that all jobs run on the correct LPAR.

For example, in “Figure 24” on page 135, the DBC for S1 is already operational and you want to replicate it to S2. See also the figures in “Figure 20” on page 107 and “Figure 21” on page 108.

To replicate the DBC by using the same DBC group and option sets

1 Review the section of the Installation System Reference Manual that describes product deployment. In this scenario, you are deploying at the LPAR level, so you start by running the $400 series of jobs.

2 Run the $$JCLCPY in the install JCL to make a copy of the original JCL to be used for deployment.

<table>
<thead>
<tr>
<th>Product code</th>
<th>Symbolic variable</th>
<th>Variable description</th>
<th>Need different value when deploying to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGC</td>
<td>DVLGCREG</td>
<td>LGC Repository</td>
<td>Different DBC Group</td>
</tr>
<tr>
<td></td>
<td>DVLGXCXF</td>
<td>LGC XCF Group Name</td>
<td>Different DBC Group</td>
</tr>
<tr>
<td>DBC</td>
<td>DVDBCSID</td>
<td>DBC SSID</td>
<td>Different DBC started task</td>
</tr>
<tr>
<td></td>
<td>DVDBCGID</td>
<td>DBC Group ID</td>
<td>Different DBC Group</td>
</tr>
<tr>
<td></td>
<td>DVDBCREP</td>
<td>DBC Repository</td>
<td>Different DBC started task</td>
</tr>
<tr>
<td></td>
<td>DVDBCSTR</td>
<td>DBC VOLSER</td>
<td>(Change as desired)</td>
</tr>
</tbody>
</table>
3 Review product-specific configuration guides for any setup required for these products to use the DBC and its agents.

4 Modify the infrastructure symbolic variables as follows:

   a In “Symbolic variables contained in $$INCINF member” on page 103, locate the rows that contain the entry "Different DBC started task" in the fourth column.

   b Use the symbolic variables in the second table column of those rows.

5 Begin running the installation jobs with the $400 series and consider the following:

   ■ (For NGL) If your original install specified to reuse NGL, then you will not have a $410VNGL job. If you do not have a $410VNGL job and you need to create a new one for this environment, contact BMC Customer Support for a sample

   ■ (For LGC) If your RTCS system registry is not shared with the original LPAR, run the $463LGCD job. If your original install specified to reuse LGC, then you will not have the $463LGCD job. If you are setting up a new LGC registry, you can customize the sample job LGC$REGD in the HLQ.LLQSAMP data set.

   ■ (Before running the $490RGIM job (if generated)) If your original install specified to migrate LGC option sets for your products, the job will copy the option sets from the previous version to the new version and then update certain values for new data set names, plan names, and so on. If your new environment does not have any option sets to migrate, contact BMC Customer Support to get a sample job that will create a new option set.

6 Run the $500 series of jobs (if generated).

7 Replicate the installation to other DB2 subsystems using the $700 series of jobs.

   See “Replicating the installation to other DB2 subsystems” on page 132.

8 Verify that the products are working correctly:

   ■ Check the DBC log and DBCPRINT for messages to ensure that the product agents are running successfully.

   ■ Invoke the product CLIST (if applicable) or run a batch job to verify that the products are working correctly.
Replicating the DBC by using a different DBC group

Use this procedure if you need to change the setup of the DBC to accommodate more than one DBC group in the same sysplex or on the same LPAR and your products require the LGC agent. This is not the default configuration, so some manual steps are required.

In “Product deployment for infrastructure components” on page 134, the DBCs for T1 and T2 are already operational and you want to replicate it to P1 using a different DBC group to isolate production from test in the same sysplex.

Using this procedure is recommended in the following situations:

- Multiple DBC subsystems are running on the same LPAR but are in different DBC groups, as shown in Figure 25 on page 142.

**Figure 25: Different DBC groups on one LPAR**

```
LPAR1

DBC TESTPLEX

DBC PRODPLEX
```
Multiple DBC subsystems are running in the same sysplex, they are in different DBC groups, and the RTCS system registry is shared, as shown in Figure 26 on page 143.

**Figure 26: Multiple DBC groups sharing one RTCS system registry**

For example, assume that you have DB2 subsystems that represent the test and production environments on the same LPAR (or in the same sysplex where the RTCS system registry is shared). You should separate the DBC subsystems that are monitoring those DB2 subsystems so that you can roll maintenance into production in a controlled manner. The test and production DBC subsystems will use separate runtime data sets, option sets, and LGC private registry datastore. Because the installation is set up to use the RTCS system registry to find the LGC private registry datastore, you need to use a special ZDBCssid DD statement in some of the installation jobs to direct TEST or PROD to its own datastore.

**Overview of using ZDBCssid to designate non-default DBC groups**

By default, when the LGC agent receives requests to retrieve option sets, it will query the RTCS system registry to find the name of the DBC group that was registered as the default group. Each individual RTCS system registry can have only one DBC group that is registered as the default.

In the situations described in Figure 25 on page 142 and Figure 26 on page 143, you can keep the TEST option sets separate from the PROD option sets by specifying the ZDBCssid designation on TEST to override the registered default DBC group of
PRODPLEX. Use the ZDBCssid designation only when you must override the default DBC group.

For DBC groups that are not registered as the default group, you will need to use the special ZDBCssid designation in:

- The DBC$STC started task procedure
- The product CLISTs
- Any jobs that execute LGCUTIL

You can designate only one DBC group with the defaultdbc element per RTCS system registry. In other words, each RTCS system registry can have only one default DBC group designated (meaning a one-to-one correlation). All other DBC groups need to use the ZDBCssid designation.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple DBC subsystems on the same LPAR</td>
<td>If the DBC subsystems are in different DBC groups, one group must be registered with the defaultdbc value. Others must use the ZDBCssid designation.</td>
</tr>
<tr>
<td>Multiple DBC subsystems in the same sysplex</td>
<td>If the DBC subsystems are in different DBC groups and your RTCS registry is shared across the LPAR, one group must be registered with the defaultdbc value. Others must use the ZDBCssid designation.</td>
</tr>
</tbody>
</table>

**Replicating the DBC (for a different DBC group) by using a ZDBCssid designation**

This procedure can help you determine when to assign a ZDBCssid designation for the non-default DBC group. For example, you could have your test environment use the ZDBCssid designation and your production environment registered as the defaultdbc. You must allocate ZDBCssid when invoking the product CLISTs in order to access the correct LGC private registry datastore for any non-default DBC subsystem.

1. Review the section of the installation system reference manual that describes product deployment. In this scenario, you will start by running the $300 series of jobs.
   
   Run the $$JCLCPY job to create a new JCL data set for this deployment scenario.

2. Review product-specific configuration guides for any set up required for these products to use the DBC and its agents.

3. Modify the infrastructure symbolic variables as follows:
a In “Symbolic variables contained in $INCINF member” on page 103, locate the rows that contain the entry "Different DBC started task," "Different DBC Group," or "Different DOMPLEX NAME" in the fourth column.

b Use the symbolic variables in the second table column of those rows.

4 Begin running the installation jobs with the $300 series of jobs, as follows:

- **(For System and SQL Performance)** Review the $310VDOM job. If your original install specified to reuse the DOM VSAM data sets, but you need to create new ones, then contact BMC Customer Support for a sample job.

- **(For LGC)** If your original install specified to reuse LGC, then you will not have a $310VLGC job. If you are setting up a new LGC registry, you can customize the ALLOC step in the sample job LGC$REGD in the HLQ.LLQSAMP data set.

5 Review “Controlling sharing of the RTCS system registry” on page 110 and “Determining the default DBC group” on page 116.

6 Assume that you have the following environment, and that you want to set up TEST to specify ZDBCssid and let PROD use the default DBC group:

<table>
<thead>
<tr>
<th>Environment</th>
<th>DBC SSID</th>
<th>DBC GROUP</th>
<th>defaultdbc</th>
<th>XCF Group</th>
<th>REGDSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST</td>
<td>TST1</td>
<td>TESTPLEX</td>
<td>No</td>
<td>TSTPLEX1</td>
<td>BMCPERF.TEST.LGCRGRY</td>
</tr>
<tr>
<td>PROD</td>
<td>PRD1</td>
<td>PRODPLEX</td>
<td>Yes</td>
<td>PRDPLEX1</td>
<td>BMCPERF.PROD.LGCRGRY</td>
</tr>
</tbody>
</table>

7 Consider the following details when running the $400 series of jobs, then start the DBC task as instructed in $450STRT:

- **(For NGL)** If your original install specified to reuse NGL, then you will not have a $410VNGL job. If you do not have a $410VNGL job and you need to create a new one for this environment, contact BMC Customer Support for a sample.

- For TEST, edit DBC$STC (the customized DBC started task in your JCL data set) to add the following DD statement to the PROC, where ssid is your DBC subsystem ID:

  ```
  //ZDBCssid DD DUMMY
  ```

  For example, in the TST1 installation on TEST, you would use ZDBCTST1, where TST1 is the DBC SSID. On other LPARS in this group, specify their DBC ID for ssid.

  **Note**

  This step is required for any DBC group that is not registered as the default DBC.
For a $450STRT job, start the DBC started task.

If your original install specified to reuse LGC, then you will not have a $463LGCD job. You can customize the sample job LGC$REGD in the HLQ.LLQSAMP data set and run the steps that are similar to the ones from $463LGCD mentioned below.

For PROD, run the LGCDFLT step from $463LGCD first to establish it as the defaultdbc.

```xml
<register>
  <defaultdbc>PRODPLEX</defaultdbc>
</register>
```

If a default DBC has already been registered, an error message will notify you. To replace the existing defaultdbc value with a new one, add the replace attribute to the XML:

```xml
<register replace="yes">
  <defaultdbc>PRODPLEX</defaultdbc>
</register>
```

For PROD, the LGCRGRP step from $463LGCD is shown in the following example:

```xml
<register>
  <dbc dbcgroup="PRODPLEX">
    <rtcsreg xcfgroup="PRDPLEX1">
      <regdsn>BMCPERF.PROD.LGCRGRY</regdsn>
    </rtcsreg>
  </dbc>
</register>
```

For TEST, the LGCDFLT step from $463LGCD is shown in the following example:

```xml
<register>
  <defaultdbc>TESTPLEX</defaultdbc>
</register>
```

You are not required to run this step, because PROD will be registered as the default DBC. Running this step will have no effect if a defaultdbc value already exists. The value will not be replaced unless the replace="yes" attribute is in the XML.

For TEST, the LGCRGRP step from $463LGCD is shown in the following example:

```xml
<register>
  <dbc dbcgroup="TESTPLEX">
    <rtcsreg xcfgroup="TSTPLEX1">
      <regdsn>BMCPERF.TEST.LGCRGRY</regdsn>
    </rtcsreg>
  </dbc>
</register>
```

8 Run the $465INIT job.
9 Edit the $470LGCR and $490RGIM jobs (if generated) and add the following DD statement after PGM=LGCUTIL where ssid is your DBC subsystem ID:

```bash
//LGCUTIL EXEC PGM=LGCUTIL,REGION=0M
//ZDBCssid DD DUMMY
```

**Note**

*(Before running the $490RGIM job (if generated)) If your original install specified to migrate LGC option sets for your products, the job will copy the option sets from the previous version to the new version and then update certain values for new data set names, plan names, and so on. If your new environment does not have any option sets to migrate, contact BMC Customer Support to get a sample job that will create a new option set.*

10 Continue running the $400 series jobs in sequential order.

11 Run the $500 series jobs (if generated).

12 Replicate the installation to other DB2 subsystems using the $700 series of jobs.

See “Replicating the installation to other DB2 subsystems” on page 132.

13 Determine which method you prefer for accessing the product option sets (as explained in “Working with option sets” on page 51):

- **(System and SQL Performance products)** If you prefer to use the DOMCLIST or SPDCLIST job in the UBMCCLIB data set, complete these steps:

  1. Edit the DOMCLIST or SPDCLIST to change the top of the PROC statement to automatically connect to your DBC subsystem.

     Having separate CLISTs for the test and production environments forces TEST to connect with TEST and PROD to connect with PROD. Use the following statement, where domplex is your DOMPLEX option set name:

     ```bash
     PROC O P('PRD=,DP=domplex') SSID()
     ```

     For example, you could use the following statement for TEST:

     ```bash
     PROC O P('PRD=,DP=TESTOPT') SSID().
     ```

     **Note**

     The DOMPLEX name should match the DOMPLEX = keyword value in the $495SDOM started task for TEST.
2 (required for any DBC group that was not registered as the default DBC)

Add the following allocation to your product CLISTs (DOMCLIST,
SPDCLIST, or DOMC) to connect to your LGC component:

```
ALLOC F(ZDBCssid) DUMMY REUSE
```

Also, add a FREE statement at the bottom, where ssid is your DBC SSID:

```
FREE F(ZDBCssid)
```

For example, TEST would use the following statements:

```
ALLOC F(ZDBCTST1) DUMMY REUSE
FREE F(ZDBCTST1)
```

3 If you want to use the same CLIST for multiple LPARS, add code similar to
the following example, and then use similar IF statements at the end of the
CLIST to free the allocation:

```
IF &SYSNAME=DB2A THEN +
  DO
    ALLOC F(ZDBCTSTA) DUMMY REUSE
  END
IF &SYSNAME=DB2B THEN +
  DO
    ALLOC F(ZDBCTSTB) DUMMY REUSE
  END
```

In this example, TSTA is the DBC ssid on LPAR DB2A and TSTB is the DBC
ssid on LPAR DB2B.

4 Execute the DOMCLIST or SPDCLIST (or hyperlink from MainView).

5 Navigate to Administration and then DOMPLEX Option Sets to edit and
customize your DOMPLEX option set.

- If you prefer to use the LGCISPF CLIST in the product UBMCCLIB or
  UBMCSAMP data set, specify the DBC ssid to use, as shown in the following
  example:

```
EX 'BMC.BMCCLIB(LGCISPF)' 'DBC(ssid)'
```

In this example, ssid is the DBC subsystem identifier. The DBC that you specify
must be active in the LPAR.

14 Verify that the products are working correctly:

- Check the DBC log and DBCPRINT for messages to ensure that the product
  agents are running successfully.
- Invoke the product CLIST (if applicable) or run a batch job to verify that the
  products are working correctly.
Avoiding ZDBCssid

If necessary, you could avoid needing the ZDBCssid designation by using a different configuration. Sample scenarios are as follows:

- **Multiple DBC subsystems are running in the same sysplex, they are in different DBC groups, and the RTCS system registry is shared.** See “Figure 26” on page 143.
  
  To avoid use of ZDBCssid in this situation, you could set up the RTCS system registry to be shared per DBC group. For more information, see “Controlling sharing of the RTCS system registry” on page 110.

- **Multiple DBC subsystems are running on the same LPAR but are in different DBC groups.** See “Figure 25” on page 142.
  
  To avoid use of ZDBCssid in this situation, use either of the following options:

  - Require the DBC subsystem registered as `defaultdbc` to always start first.
  
  - Set up a dedicated DBC group to be registered with the `defaultdbc` parameter, include only the LGC agent in it, and always require the DBC group to start before the other DBC subsystems.

Use the following procedure if you want to create a DBC that will be in a different DBC group from the one registered as the `defaultdbc`:

1. Review the section of the installation system reference manual that describes product deployment. In this scenario, you will start by running the $300 series of jobs.
   
   Run the `$JCLCPY` job to create a new JCL data set for this deployment scenario.

2. Review product-specific configuration guides for any set up required for these products to use the DBC and its agents.

3. Modify the infrastructure symbolic variables as follows:

   a. In “Symbolic variables contained in `$INCINF` member” on page 103, locate the rows that contain the entry "Different DBC started task," "Different DBC Group," or "Different DOMPLEX NAME" in the fourth column.

   b. Use the symbolic variables in the second table column of those rows.

4. Begin running the installation jobs with the $300 series of jobs.

   *(For System and SQL Performance)* Review the $310VDOM job. If your original install specified to reuse the DOM VSAM data sets, but you need to create new ones, then contact BMC Customer Support for a sample job.
Do not run the $310VLGC job. In this scenario, the defaultdbc will be the only DBC that has an active LGC agent and its datastore will contain option sets for all the products.

5 Consider the following when running the $400 series of jobs:

— *(For NGL)* If your original install specified to reuse NGL, then you will not have a $410VNGL job. If you do not have a $410VNGL job and you need to create a new one for this environment, contact BMC Customer Support for a sample.

For the $450STRT job, start the DBC started task that was registered as the defaultdbc. Make sure that the LGC agent defined to the task is active. Then start the DBC for this new environment.

— *(For LGC)* Do not run the $463LGCD job. It is not needed in this scenario because the task uses the LGC registry from the defaultdbc.

Note

- Do not run any of the LGC steps in the $465INIT job. These steps define the LGC agent to the DBC. This is not necessary for this scenario because the task uses the LGC agent from the defaultdbc.

- Do not run $470LGCR. This job registers the LGC templates, which was already done for the LGC in the defaultdbc. The first step of this job, LGCLIST, will list all DBC groups and LGC registries and indicate which DBC is registered as the defaultdbc. You could run this step to verify the defaultdbc.

- *(Before running the $490RGIM job (if generated))* If your original install specified to migrate LGC option sets for your products, the job will copy the option sets from the previous version to the new version and then update certain values for new data set names, plan names, and so on. If your new environment does not have any option sets to migrate, contact BMC Customer Support to get a sample job that will create a new option set.

6 Run the $500 series of jobs (if generated).

7 Replicate the installation to other DB2 subsystems using the $700 series of jobs. See “Replicating the installation to other DB2 subsystems” on page 132.

8 Verify that the products are working correctly:
Check the DBC log and DBCPRINT for messages to ensure that the product agents are running successfully.

Invoke the product CLIST (if applicable) or run a batch job to verify that the products are working correctly.

If you need help with a specific situation or you need more information, contact BMC Customer Support.

Replicating the installation to another sysplex on non-shared DASD

Use this procedure to replicate the installation to another sysplex on non-shared DASD.

For example, in “Figure 24” on page 135, the DBCs for SYSPLEX X1 are already operational and you want to replicate it to SYSPLEX X3.

1 Review the section of the Installation System Reference Manual that describes product deployment. In this scenario, you are deploying at the SYSPLEX level, so you will start with running the $300 series of jobs.

2 Run the $$JCLCPY job to create a new JCL data set for this deployment scenario.

3 If necessary, use your own process to transport the runtime data sets to the other environment.

4 Modify the infrastructure symbolic variables as follows:
   a In “Symbolic variables contained in $$INCINF member” on page 103, locate the rows that contain the entry "Different DBC started task," "Different DBC Group," or "Different DOMPLEX NAME" in the fourth column.
   b Use the symbolic variables in the second table column of those rows.

5 Begin running the installation jobs with the $300 series of jobs, as follows:

   (For System and SQL Performance) Review the $310VDOM job. If your original install specified to reuse the DOM VSAM data sets, but you need to create new ones, then contact BMC Customer Support for a sample job.

   (For LGC) If your original install specified to reuse LGC, then you will not have a $310VLGC job. If you need to create a new LGC registry for this environment,
you can customize the ALLOC step in the sample job LGCS$REGD in the
HLQ.LLQSAMP data set.

6 Run the $400 series of jobs and consider the following:

■ (For NGL) If your original install specified to reuse NGL, then you will not
have a $410VNGL job. If you do not have a $410VNGL job and you need to
create a new one for this environment, contact BMC Customer Support for a
sample.

■ (For LGC) If your RTCS system registry is not shared with the original LPAR,
run the $463LGCD job. If your original install specified to reuse LGC, then you
will not have the $463LGCD job. If you are setting up a new LGC registry, you
can customize the sample job LGCS$REGD in the HLQ.LLQSAMP data set.

■ (Before running the $490RGIM job (if generated)) If your original install
specified to migrate LGC option sets for your products, the job will copy the
option sets from the previous version to the new version and then update
certain values for new data set names, plan names, and so on. If your new
environment does not have any option sets to migrate, contact BMC Customer
Support to get a sample job that will create a new option set.

7 Run the $500 series of jobs (if generated).

8 Replicate the installation to other DB2 subsystems using the $700 series of jobs.

See “Replicating the installation to other DB2 subsystems” on page 132.

9 Verify that the products are working correctly:

■ Check the DBC log and DBCPRINT for messages to ensure that the product
agents are running successfully.

■ Invoke the product CLIST (if applicable) or run a batch job to verify that the
products are working correctly.
Index

$INCINF 112, 118, 138
$INCUSR 118, 132, 138
$310VZMC job 111
$463LGCD job 140, 146
$465INIT job 93, 117, 118, 123
$470LGCR 149
$480INIT job 117, 118, 123
$490RGIM job 118, 123
$S00JCL 133

A

ABOUT command 60
AF_UNIX 21
agents
  deploying maintenance 117, 124
  DOM 122, 125
  GUD 122, 125
  LGC 122, 125
  NGL 122, 125
ALLOW_SAF_RC4 element 38, 45
ALTGROUP command 39
ALTUSER command 39
APPTUNE for DB2 13
ARCHIVE LOGSET command 101
authorizing DBC started task 38

B

BMC Subsystem Optimizer (for zEnterprise) 13
BMC Workbench (for DB2) 106
BMC Workbench for DB2 13
BPXPRMxx 21
BRD commands
  REFRESH 122
  START 127
  STOP 125

C

CANCEL command 60
CHKZMC1 step 111
COMMAND command 61
COMMAND element 45
Command line commands in LGC 60
commands
  DBC 33
  DIAG 33
  NGL commands 100
  RESETPRINT 33
  RESETTRACE 33
  STOP 33
  TRACEFLUSH 33
  TRACEOFF 33
  TRACEON 33
COMPONENT element 44
config element 73
configuration of products 47
CONTEXT element 43
CONTEXTS element 43
conventions, documentation 11
create element 66
creating option sets 53
cross-system coupling facility (XCF) groups 112

D

Data Collector
  migrating 131
Data Collector commands
    STATUS command 102
DATACLAS element 31
DB2 Component Services (DBC)
deployment 105
started task 105, 113, 114
default DBC group 116
description 14
group names 142
infrastructure components 13
products supported 13
replicating 142
replicating subsystems 140
reusing 130
started task 105, 113, 114
user ID 18
DB2 objects 132
DB2 Product Configuration (LGC)
deployment 105
interface 52
Command line commands 60
description 15
drop-down menus 48, 50
infrastructure components 13
interview overview 48
menu bar 50
online Help 51
Product Option Sets panel 48
products supported 13
reusing 131
sections 48
DB2 Production Configuration (LGC)
option sets 47
overview 47
DBC
stopping 32
See also DB2 Component Services (DBC)
DBC commands
DIAG 33
START 128
DISPAGENTS 33
DISPDSNS 33
RESETPRINT 33
RSETTRACE 33
STOP 33, 127
TRACEFLUSH 33
TRACEOFF 33
TRACEON 33
DBC started task
authorizing 38
DBCPARMS 21
DBCPRINT 23
DBCSECUR 23
DOMPARMS 21
DOMTMPLT 23
JCL 21
starting 21
STEPLIB 23
SYSPRINT 23
user ID 18
DBC subsystem
replicating 140
starting 21
stopping 21
working with 21
DBC$STC started task 113, 114, 144
DBCDSNS job 124
DBCPARMS DD statement 21, 37
DBCPARMS parameters 21, 23
DBCPRINT DD statement 23
DBCSECUR DD statement 23, 37, 41
DBCSECUR element 42
ddbname element 74
default DBC group 116
default DBC group name 143
defaultdbc element 75
deleting option sets 58
deployment environments 105
deregister element 66
description element 76
DIAG command 33
DISPLAY LOGSET command 101
documentation information 9
DOM agents
refreshing 122
stopping 125
installing 131
starting 127
DOM commands
REFRESH 123
START 127
stop 125
DOMCLIST 52
DOMINIT step 120
DOMPARMS DD statement 21
DOMPLEX option sets 56
DOMTMPLT DD statement 23
DOMVARS data set 106
DPRREPOS element 29
drop-down menus in LGC 48, 50
dsn element 76
DUMP command 100

E
editing option sets 54, 55
electronic documentation 9
export element 67

F
file element 76
files element 77
FILTDEF command 59, 60
filtering option sets 58
FILTINV command 59, 60
FILTMOD command 59, 60
FILTOFF command 59, 60
FILTREQ command 59, 60
FMID
  ZDBCxxx 14
  ZLGCxxx 15
  ZNGLxxx 16
FMID element 32

G
GENHELP command 60
GROUP element 27
GUD agents
  refreshing 122
  stopping 125
  starting 127
GUD commands
  STARTALL 123, 127
  STOPALL 123, 125
  STARTALL 123, 127
  STOPALL 123, 125
GUDINIT step 119

H
Help
  online 9
HELP command 60

I
ICHERCDE macro 39
ICHRFRTB macro 40
IGNORE element 31
import element 68
Index Term 96, 140, 151
installation user ID 17
instance element 79

L
LGC. See DB2 Product Configuration
LGC agents
  refreshing 122
  stopping 125
  starting 127
LGC commands
  COMMAND 61
  REFRESH 61, 122
  REGSTAT 61
  RELQUES 61
  STATUS 61
  STOP 61, 125
  interface 60
  START 127
  STOP 61, 125
  STORAGE 61, 63
  TRACE 61, 62
LGC$LIST 116
LGCDFLT step 146
LGCCINIT step 120
LGCSI PF 53
LGCRGRP step 146
lgcutil element 65
LGCU TIL step 116
list element 69
LIST LOGSET command 100
load libraries
  removing references 123
  changing references 118
LOADLIB element 117, 118
LPAR requirements 105
M
maintenance 117, 124
MainView for DB2 - Data Collector 13
MainView for WebSphere MQ 13
MainView Transaction Analyzer 13
MAXSOCKETS 21
member element 79
menu bar in LGC 48, 50
MGMTCLAS element 31
migrate element 69
MODE element 28
MVINIT step 120
MVL 118, 123, 124

N
name element 80
NAME element 29
newdescription element 80
newname element 81
Next Generation Logger (NGL) started task 19, 93
commands 100
deployment 105
description 16
infrastructure components 13
NGLARCH started task 106
products supported 13
registries 95
reusing 131
started task 19, 93
user ID 19
NGL. See Next Generation Logger (NGL)
NGL agents
stopping 125
refreshing 122
starting 127
NGL commands
DUMP 100
REFRESH 100, 123
START 100, 127
STOP 100, 126
START 100, 127
STATUS LOGSET command 101
STOP 100, 126
NGLAGJV step 120
NGLARCH started task 19, 93, 106
NGLINIT step 93, 120

O
OMVS initialization 21
OMVS segment 39
online Help 9
online Help in LGC 51
option sets
interface 52
name guidelines 52
creating 53
creating section defaults 56
default values 55
deleting 58
editing 54, 55
filtering 58
LGC 47
renaming 57
OPTIONS element 26
optionset element 81
optionsets element 83
OSZ0240I message 111

P
PACLOG for DB2 13
PIID element 32
PMDHIST data set 106
Pool Advisor for DB2 13
prodcode element 83
PRODHELP command 60
product configuration 47
product element 84
PRODUCT element 32
Product Option Sets panel in LGC 48
products element 85
prodvrsn element 86
publications, related 9

R
RACF resource class 39
RDEFINE FACILITY commands 40
Recovery Management for DB2 13
RECOVERY MANAGER for DB2 13
REFRESH command 61, 100
regdsn element 87
REGIOD 118, 123
register element 70
REGISTRY DD statement 95
REGSTAT command 61
related publications 9
RELQUES command 61
rename element 71
renaming option sets 57
RESETPRINT command 33
RESETTRACE command 33
RESOURCE_CLASS element 44
RESOURCE_NAME element 42
REXX Exec 101
RTCS. See Runtime Component System (RTCS)
RTCS system registry 111, 116, 140, 143
rtcsreg element 87
Runtime Component System (RTCS)
deployment 105
description 16
infrastructure components 13
level required 110
products supported 13
reusing 129
system registry 111
runtime data sets
replicating 134
SAVE command 60
SAVEAS command 60
sections in LGC 48
security 36
SETROPTS command 40
SMFID element 43
solution element 88
solutions element 89
SPDCLIST 52
SQL Performance for DB2 13
SSID element 26
START command 100, 127
started task
NGL 93
NGLARCH 106
DBC 105, 114
STATUS command 61
Data Collector commands 102
STATUS LOGSET command
NGL commands 101
STEPLIB DD statement 23
STOP command 33, 61, 100, 124, 125
stopping DBC 32
STORAGE command 61, 63
STORCLAS element 30
SUBSYS element 45
SUBSYSTYPE element 29
Subzero 13
syntax statement conventions 10
SYSPRINT DD statement 23
System Authorization Facility (SAF) interface 37
System Performance for DB2 13
T
technology components
  DB2 Component Services 14
  DB2 Product Configuration 15
  Next Generation Logger (NGL) 16
  Runtime Component System (RTCS) 16
template element 90
templates element 90
TO_VALUE element 43
TRACE command 61, 62
TRACEFLUSH command 34
TRACEOFF command 33
TRACEON command 34
U
update element 71
user ID 19
user IDs
  installation 17
  authorizing 39
V
VOLUMES element 30
W
WLM element 28
WTOUPPERCASE element 29
X

XCF groups 112
XCFGROUP element 28, 114

Z

ZDBssid 143, 149
ZLGCxxx 15