Fourth Edition, Softcopy Only (February 2006)

This edition applies to Version 8 of IBM DB2 Universal Database for z/OS (DB2 UDB for z/OS), product number 5625-DB2, and to any subsequent releases until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

This softcopy version is based on the printed edition of the book and includes the changes indicated in the printed version by vertical bars. Additional changes made to this softcopy version of the book since the hardcopy book was published are indicated by the hash (#) symbol in the left-hand margin. Editorial changes that have no technical significance are not noted.

This and other books in the DB2 for z/OS library are periodically updated with technical changes. These updates are made available to licensees of the product on CD-ROM and on the Web (currently at www.ibm.com/software/data/db2/zos/library.html). Check these resources to ensure that you are using the most current information.

© Copyright International Business Machines Corporation 1994, 2006. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

About this book ........................................... vii
Who should read this book .......................... vii
Terminology and citations ........................... vii
Accessibility ........................................ viii
How to send your comments ....................... viii

Summary of changes to this book ............... ix

Chapter 1. Introduction to DB2 data sharing ........................................... 1
Advantages of DB2 data sharing ..................... 1
  Improved data availability .......................... 1
  Extended processing capacity ...................... 2
  Configuration flexibility ............................ 4
  Higher transaction rates ............................ 8
Leaves application interface unchanged .......... 8
How DB2 data sharing works ....................... 8
  How DB2 protects data consistency ............ 8
  How an update happens ............................ 10
Using DB2 data sharing .............................. 14
  Enabling DB2 data sharing ....................... 14
  Connecting to a data sharing group .......... 15
  Administering a database ....................... 15
  Operating a data sharing group ............... 16
Software and hardware requirements ........... 19
  Software .......................................... 20
  Hardware ......................................... 20
Storage estimates .................................... 20

Chapter 2. Planning for DB2 data sharing ........................................... 21
Planning for DB2 data sharing in a Parallel Sysplex .................. 21
  Parallel Sysplex components and requirements .... 21
  Connectivity requirements ....................... 26
  Data sharing naming conventions ............... 27
    Data sharing group names ..................... 27
    Member names .................................. 28
    IRLM names .................................... 30
    Coupling facility structure names ............ 31
    Naming recommendations ...................... 32
    Naming example for DB2 data sharing ........ 34
Planning for availability .......................... 35
  Automatic restart of z/OS ....................... 36
  Coupling facility availability .................. 37
  Duplexing structures ............................. 42
  DB2 resource availability considerations ...... 44
Planning for WebSphere coordinated XA transactions .............. 45
Estimating storage ................................... 46
  General information about coupling facility storage ........................................ 47
  Group buffer pool sizes .......................... 48
  Lock structure size .............................. 54
  SCA size ........................................ 55
Changing structure sizes .................................. 56
Estimating additional storage for IRLM ............ 57
Storage for DB2 objects ............................ 58
# Before you enable DB2 data sharing ............... 59
  Deciding if merging is the right thing to do ....... 59
  Connecting IMS and CICS ....................... 60
  Registering command prefixes and member group attachment name .......... 61
  Specifying the group attachment name .......... 61
  Increasing the size of the BSDS ............... 63
  Increasing the size of the SYSLGRNX table space .... 63
Application design planning ........................ 64
  Using the NO CACHE option when creating sequences .................. 64
  Applications using CICSplex System Manager .... 64
  Migrating transactions that have ordered dependencies ........... 67
  Binding plans and packages if you are moving to a new machine ....... 68

Chapter 3. Installing and enabling DB2 data sharing ...................... 69
Choosing parameters for members ..................... 69
  The scope and uniqueness of DB2 subsystem parameters .................. 70
  DSNHDECP parameters .................................. 75
Creating a data sharing group ...................... 75
  Recommended approach for moving to DB2 data sharing ............... 75
  Sharing DB2 libraries .................................. 76
  Ensuring that installation jobs access the correct JCL procedures .......... 77
  Establishing system affinity for installation jobs .............. 77
Installing a new data sharing group ............... 77
  Renaming a member .................................. 78
  Tasks that require an IPL .................................. 78
  Tasks at enable time .................................. 79
Enabling DB2 data sharing .......................... 80
Adding new members .................................. 81
Merging existing DB2 data into the group ........ 83
  Merging subsystems .................................. 83
  Merging data ........................................ 83
Testing the data sharing group ..................... 89
  Testing group buffer pool caching .................. 89
  Testing global lock serialization .................. 90
  Testing concurrency .................................. 90
Testing Sysplex query parallelism ................. 90
Updating subsystem parameters for a member .... 91
Migrating an existing data sharing group to a new release .......... 91
  Mixed releases in a data sharing group .............. 92
  Procedure to migrate the data sharing group .... 99
Falling back and remigrating ........................ 100
Falling back .......................................... 100
Falling back and disabling DB2 data sharing .......... 101
Changing the size of the lock structure . . . 234
Tuning group buffer pools . . . . . . . . . 236
Assigning page sets to group buffer pools . 237
Inter-DB2 interest and GBP-dependency . . 238
Using P-locks . . . . . . . . . . . . . . . . . . . 243
Read operations . . . . . . . . . . . . . . . . . . . 246
Write operations . . . . . . . . . . . . . . . . . . . 248
Group buffer pool thresholds . . . . . . . . 255
Monitoring group buffer pools . . . . . . . 258
Determining the correct size and ratio . . . 263
Changing group buffer pools . . . . . . . . 269
Access path selection in a data sharing group . 272
Effect of member configuration on access path selection . . . 272
How EXPLAIN works in a data sharing group 273
Maintaining in-memory statistics in data sharing 273

Appendix. DB2 and IRLM names . . . 275
DB2 group names . . . . . . . . . . . . . . . . . 275
DB2 member names . . . . . . . . . . . . . . . . 275
IRLM names . . . . . . . . . . . . . . . . . . . . 276

Notices . . . . . . . . . . . . . . . . . . . . . 279
Programming interface information . . . . . 280
Trademarks . . . . . . . . . . . . . . . . . . . . 281

Glossary . . . . . . . . . . . . . . . . . . . . 283

Bibliography . . . . . . . . . . . . . . . . . . 317

Index . . . . . . . . . . . . . . . . . . . . . . 325
About this book

This book is the main source of information about DB2® data sharing. You can use it to educate yourself about DB2 data sharing and to do many of the tasks that are associated with DB2 data sharing.

However, there are many tasks that are associated with DB2 data sharing, especially those of setting up the hardware and software environment for the Parallel Sysplex®, that require the use of other product libraries, such as z/OS. If you are installing DB2 and have plans to use data sharing capabilities, use this book with [DB2 Installation Guide] to do initial planning and develop your installation strategy. You can find detailed installation procedures in [DB2 Installation Guide]. Exceptions to, and deviations from, the standard procedures are noted in this book.

Important
In this version of DB2 UDB for z/OS, the DB2 Utilities Suite is available as an optional product. You must separately order and purchase a license to such utilities, and discussion of those utility functions in this publication is not intended to otherwise imply that you have a license to them. See Part 1 of [DB2 Utility Guide and Reference] for packaging details.

Who should read this book

This book is primarily intended for system and database administrators who are responsible for planning and implementing DB2 data sharing. Many of the task descriptions in this book assume that the user is already familiar with administering DB2 in a non-DB2 data sharing environment. See [DB2 Administration Guide] for any concepts not explained in this book.

Terminology and citations

In this information, DB2 Universal Database™ for z/OS® is referred to as "DB2 UDB for z/OS." In cases where the context makes the meaning clear, DB2 UDB for z/OS is referred to as "DB2." When this information refers to titles of books in this library, a short title is used. (For example, "See DB2 SQL Reference" is a citation to IBM® DB2 Universal Database for z/OS SQL Reference.)

When referring to a DB2 product other than DB2 UDB for z/OS, this information uses the product’s full name to avoid ambiguity.

The following terms are used as indicated:

DB2 Represents either the DB2 licensed program or a particular DB2 subsystem.

OMEGAMON

Refers to any of the following products:

- IBM Tivoli OMEGamon XE for DB2 Performance Expert on z/OS
- IBM Tivoli OMEGamon XE for DB2 Performance Monitor on z/OS
- IBM DB2 Performance Expert for Multiplatforms and Workgroups
- IBM DB2 Buffer Pool Analyzer for z/OS
C, C++, and C language

Represent the C or C++ programming language.

CICS®

Represents CICS Transaction Server for z/OS or CICS Transaction Server for OS/390®.

IMS™

Represents the IMS Database Manager or IMS Transaction Manager.

MVS™

Represents the MVS element of the z/OS operating system, which is equivalent to the Base Control Program (BCP) component of the z/OS operating system.

RACF®

Represents the functions that are provided by the RACF component of the z/OS Security Server.

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products. The major accessibility features in z/OS products, including DB2 UDB for z/OS, enable users to:

- Use assistive technologies such as screen reader and screen magnifier software
- Operate specific or equivalent features by using only a keyboard
- Customize display attributes such as color, contrast, and font size

Assistive technology products, such as screen readers, function with the DB2 UDB for z/OS user interfaces. Consult the documentation for the assistive technology products for specific information when you use assistive technology to access these interfaces.

Online documentation for Version 8 of DB2 UDB for z/OS is available in the Information management software for z/OS solutions information center, which is an accessible format when used with assistive technologies such as screen reader or screen magnifier software. The Information management software for z/OS solutions information center is available at the following Web site:

http://publib.boulder.ibm.com/infocenter/dzichelp

How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 UDB for z/OS documentation. You can use the following methods to provide comments:

- Send your comments by e-mail to db2pubs@vnet.ibm.com and include the name of the product, the version number of the product, and the number of the book. If you are commenting on specific text, please list the location of the text (for example, a chapter and section title, page number, or a help topic title).
- You can also send comments from the Web. Visit the library Web site at:

  www.ibm.com/software/db2zos/library.html

  This Web site has a feedback page that you can use to send comments.
- Print and fill out the reader comment form located at the back of this book. You can give the completed form to your local IBM branch office or IBM representative, or you can send it to the address printed on the reader comment form.
Summary of changes to this book

The principal additions to this book are:

- More detailed information on the procedure for moving data.
- Information about using the NO CACHE option when creating sequences.
- Recommendations on planning your data sharing configuration for the highest availability and optimal network connectivity.
- Information on recovering data in XA transactions.
Chapter 1. Introduction to DB2 data sharing

The data sharing function of the licensed program DB2 UDB for z/OS enables multiple applications to read from, and write to, the same DB2 data concurrently. The applications can run on different DB2 subsystems residing on multiple central processor complexes (CPCs) in a Parallel Sysplex®.

A Sysplex is a group of z/OS systems that communicate and cooperate with one another using specialized hardware and software. They are connected and synchronized through a Sysplex Timer® and enterprise systems connection channels. A Parallel Sysplex is a Sysplex that uses one or more coupling facilities, which provide high-speed caching, list processing, and lock processing for any applications on the Sysplex.

A collection of one or more DB2 subsystems that share DB2 data is called a data sharing group. DB2 subsystems that access shared DB2 data must belong to a data sharing group.

A DB2 subsystem that belongs to a data sharing group is a member of that group. Each member can belong to one, and only one, data sharing group. All members of a data sharing group share the same DB2 catalog and directory, and all members must reside in the same Parallel Sysplex. Currently, the maximum number of members in a data sharing group is 32.

All members of a data sharing group can read and update the same DB2 data simultaneously. Therefore, all data that different members of the group can access must reside on shared disks.

Some capabilities described in this book can be used in both data sharing and non-data sharing environments. This book uses the term data sharing environment to describe a situation in which a data sharing group has been defined with at least one member. In a non-data sharing environment, no group is defined.

This chapter describes the following topics:

- “Advantages of DB2 data sharing”
- “How DB2 data sharing works” on page 8
- “Using DB2 data sharing” on page 14
- “Software and hardware requirements” on page 19

Advantages of DB2 data sharing

DB2 data sharing improves the availability of DB2 data, extends the processing capacity of your system, provides more flexible ways to configure your environment, and increases transaction rates. You do not need to change the SQL in your applications to use data sharing, although you might need to do some tuning for optimal performance.

Improved data availability

More users demand access to DB2 data every hour, every day. Data sharing helps you meet your service objectives by improving data availability during both planned and unplanned outages.
Because data sharing provides multiple paths to data, a member can be down, and applications can still access the data through other members of the data sharing group. As Figure 1 illustrates, when an outage occurs and one member is down, transaction managers are informed that the member is unavailable, and they can direct new application requests to another member of the group.

Extended processing capacity

As you move more data processing onto DB2, your processing needs can exceed the capacity of a single system. This section describes how data sharing meets your ever-increasing capacity needs.

Without DB2 data sharing

Without DB2 data sharing, you have the following options for addressing increased capacity needs:

- Copy the data, or split the data between separate DB2 subsystems.
  This approach requires that you maintain separate copies of the data. There is no communication between or among DB2 subsystems, and no shared DB2 catalog or directory.
- Install another DB2 subsystem and rewrite applications to access the data as distributed data.
  This approach might relieve the workload on the original DB2 subsystem, but it requires changes to your applications and has performance overhead of its own. Nevertheless, if DB2 subsystems are separated by great distance or DB2 needs to share data with another system, the distributed data facility is still your only option.
- Install a larger processor and move the data and applications to that machine.
This option can be expensive. In addition, this approach demands that your system come down while you move to the new machine.

**With DB2 data sharing**

With DB2 data sharing, you get the following benefits:

*Support for incremental growth:* A Parallel Sysplex can grow incrementally, allowing you to add processing power in granular units and in a non-disruptive manner. The coupling technology of Parallel Sysplex along with the additional CPU power results in more throughput for users’ applications. You no longer need to manage multiple copies of data, and all members of the data sharing group share a single DB2 catalog and directory.

*Workload balancing:* DB2 data sharing provides workload balancing so that when the workload increases or you add a new member to the group, you do not need to distribute your data or rewrite your applications. DB2 data sharing is unlike the partitioned-data approach to parallelism (sometimes called *shared-nothing* architecture), in which a one-to-one relationship exists between a database management system (DBMS) and a segment of data. When you add a new DB2 subsystem onto another central processor complex (CPC) in a data sharing environment, applications can access the same data through the new member just as easily as through any of the existing members.

DB2 works closely with the Workload Manager (WLM) component of z/OS to ensure that incoming requests are optimally balanced across the members of a data sharing group. All members of the data sharing group have the same concurrent and direct read-write access to the data.

*Capacity when you need it:* A data sharing configuration can handle peak work loads (such as end-of-quarter processing) well. You can have data sharing members in reserve, bring them online to handle peak loads, and then stop them when the peak passes.

*More capacity to process complex queries*  
*Sysplex query parallelism* enables DB2 to use all the processing power of the data sharing group to process a single query. For complex data analysis or decision support, Sysplex query parallelism is a scalable solution. Because the data sharing group can grow, you can put more power behind queries even as those queries become increasingly complex and run on larger and larger sets of data.

[Figure 2 on page 4] shows that all members of a data sharing group can participate in processing a single query.
Figure 2 is a simplification of the concept—several members can access the same physical partition. To take full advantage of parallelism, use partitioned table spaces.

**Figure 2.** Query processed in parallel by members of a data sharing group. Different members process different partitions of the data.

**Configuration flexibility**

Data sharing lets you configure your system environment much more flexibly.

As Figure 3 on page 5 shows, you can have more than one data sharing group on the same Parallel Sysplex. You might, for example, want one group for testing and another group for production data. This example also shows a single, non-data sharing DB2 subsystem.
**Flexible operational systems**

Figure 4 on page 6 shows how, with data sharing, you can have query user groups and online transaction user groups on separate z/OS images. This configuration lets you tailor each system specifically for that user set, control storage contention, and provide predictable levels of service for that set of users. Previously, you might have had to manage separate copies of data to meet the needs of different user groups.
Flexible decision support systems

Figure 5 on page 7 shows two different decision support configurations. A typical configuration separates the operational data from the decision support data. Use this configuration when the operational system has environmental requirements that are different from those of the decision support system. For example, the decision support system might be in a different geographical area, or security requirements might be different for the two systems.

DB2 offers another option—a combination configuration. A combination configuration groups your operational and decision support systems into a single data sharing group and offers the following advantages:

- You can occasionally join decision support data and operational data using SQL.
- You can reconfigure the system dynamically to handle fluctuating workloads. (You can dedicate CPCs to decision support processing or operational processing at different times of the day or year.)
- You can reduce the cost of computing:
  - The infrastructure used for data management is already in place.
  - You can create a prototype of a decision support system in your existing system, and then add processing capacity as the system grows.
If you want a combination system configuration, you must separate decision support data from operational data as much as possible. Buffer pools, disks, and control units that you use to decide on a support system should be separate from those that you use in your operational system. This separation greatly minimizes any negative performance impact on the operational system.

If you are unable to maintain the needed level of separation, or if you have separated your operational data for other reasons, such as security, using a separate decision support system is your best option.

**Flexibility to manage shared data**

DB2 data sharing can simplify the management of applications that must share data, such as a common customer table. Perhaps these applications were split in the past for capacity or availability reasons. But with the split architecture, the shared data must be synchronized across multiple systems (that is, by replicating data).

DB2 data sharing gives you the flexibility to configure these applications to access a single data sharing group. It also allows you to maintain a single copy of the

---

*Figure 5. Flexible configurations for decision support. Data sharing lets you configure your systems in the way that works best in your environment.*
shared data that can be read and updated by multiple systems with good performance. This is an especially powerful option when data centers are consolidated.

**Higher transaction rates**

Data sharing gives you opportunities to put more work through the system. As Figure 6 illustrates, you can run the same application on more than one member to achieve transaction rates that are higher than possible on a single DB2 subsystem.

![Figure 6. Data sharing enables growth. You can move some of your existing DB2 workload onto another central processor complex (CPC).](image)

**Leaves application interface unchanged**

Your investment in people and skills is protected because existing SQL interfaces and attachments remain intact when sharing data.

You can bind a package or plan on one member of a data sharing group and run that package or plan on any other member of the group.

**How DB2 data sharing works**

This section provides an overview of how shared data is updated and how DB2 protects the consistency of that data. It also introduces operational and database design considerations for the data sharing environment.

**How DB2 protects data consistency**

Applications can access data from any member of a data sharing group. Many members can potentially read and write the same data. DB2 UDB for z/OS uses special data sharing locking and caching mechanisms to ensure data consistency across the applications.
When multiple members of a data sharing group open the same table space, index space, or partition, and at least one of them opens it for writing, the data is said to be of inter-DB2 read/write interest to the members. (Sometimes called *inter-DB2 interest.*) To control access to data that is of inter-DB2 interest, whenever the data is changed, DB2 caches it in a storage area that is called a *group buffer pool.*

When there is inter-DB2 read/write interest in a particular table space, index, or partition, it is dependent on the group buffer pool, or *GBP-dependent* (group buffer pool-dependent).

You define group buffer pools by using coupling facility resource management (CFRM) policies. For more information about these policies, see *z/OS MVS Setting Up a Sysplex.*

As shown in [Figure 7](#), a mapping exists between a group buffer pool and the buffer pools of the group members. For example, each member has a buffer pool named BP0. For data sharing, you must define a group buffer pool (GBP0) in the coupling facility that maps to each member’s buffer pool BP0. GBP0 is used for caching the DB2 catalog and directory table spaces and indexes, and any other table spaces, indexes, or partitions that use buffer pool 0.

Although a single group buffer pool cannot reside in more than one coupling facility (unless it is duplexed), you can put group buffer pools in more than one coupling facility.

![Figure 7. Relationship of member buffer pools to the group buffer pool.](image)

When you change a particular page of data, DB2 caches that page in the group buffer pool. The coupling facility invalidates any image of the page in the buffer pools associated with each member. Then, when a request for that same data is subsequently made by another member, that member looks for the data in the group buffer pool.
Performance options to fit your application's needs: By default, DB2 caches updated data, but you also have the options of caching all or none of your data. There is even an option especially for large object (LOB) data.

How an update happens

Let us follow a page of data as it goes through the update process. The most recent version of the data page is shaded in the following illustrations. This scenario assumes that the group buffer pool is used for caching changed data that is duplexed for high availability. Duplexing is the ability to write data to two instances of a structure: in this case, a primary group buffer pool and a secondary group buffer pool.

Suppose, as shown in Figure 8, that an application issues an UPDATE statement from DB2A. The data that is being updated does not already reside in either the member's own buffer pool or the group buffer pool; therefore, DB2A retrieves the data from disk and updates the data in its own buffer pool. Simultaneously, DB2A gets the appropriate locks to prevent another member from updating the same data at the same time. After the application commits the UPDATE, DB2A releases the corresponding locks. The changed data page remains in DB2A's buffer pool.

Next, suppose another application, that runs on DB2B, needs to update the same data page (see Figure 9 on page 11). DB2 dynamically detects inter-DB2 interest in the page set, so DB2A writes the changed data page to the group buffer pools (both primary and secondary). DB2B then retrieves the data page from the primary group buffer pool.

Figure 8. An application running on DB2A reads data from disk and updates it
After the application that runs on DB2B commits the UPDATE, DB2B moves a copy of the data page into the group buffer pools. This invalidates the data page in DB2A’s buffer pool (see Figure 10 on page 12). Cross-invalidation occurs from the primary group buffer pool.

Figure 9. DB2B updates the same data page. When DB2B references the page set, it gets the most current version of the data from the primary group buffer pool.
Now, as shown in Figure 11 on page 13, when DB2A attempts to reread the data, it detects that the data page in its own buffer pool is invalid. Therefore, it reads the latest copy of the data from the primary group buffer pool.

Figure 10. The updated data page is written to the group buffer pools and the data page in DB2A’s buffer pool is invalidated.
How DB2 writes changed data to disk

Periodically, DB2 must write changed pages from the primary group buffer pool to disk. This process is called castout. The member that is responsible for casting out the changed data uses its own address space because no direct connection exists from a coupling facility to disk (Figure 12 on page 14). The data passes through a private buffer, not through the DB2 buffer pools.
When a group buffer pool is duplexed, data is not cast out from the secondary group buffer pool to disk. After a set of pages is written to disk from the primary group buffer pool, DB2 deletes those pages from the secondary group buffer pool.

Using DB2 data sharing

Because DB2 data sharing does not affect the application interface, it does not create additional work for application programmers and end users. However, system programmers, operators, and database administrators must perform additional tasks in a data sharing environment. Those tasks are briefly described in this section, including:

- “Enabling DB2 data sharing”
- “Connecting to a data sharing group” on page 15
- “Administering a database” on page 15
- “Operating a data sharing group” on page 16

Enabling DB2 data sharing

You must plan a naming convention before enabling data sharing on the first member (the originating member) of the group. Because names in the Parallel Sysplex and names in the data sharing group must be unique, you must have a naming convention before you create the group. Not only must shared data objects have unique names, but you must create a unique name for every group resource. See “Data sharing naming conventions” on page 27 for more information about naming.

The originating member’s DB2 catalog becomes the catalog for all the members of the data sharing group. You add additional members to the group as new installations, and those members use the originating member’s DB2 catalog.

If you have data from existing DB2 subsystems that you want the group to share, you must merge the catalog definitions for that data into the group catalog. You
must also ensure that all members of the group can access the data. DB2 does not provide a way to merge members’ catalogs automatically.

**Connecting to a data sharing group**

Applications can communicate with a data sharing group by using either Transmission Control Protocol/Internet Protocol (TCP/IP) or Systems Network Architecture (SNA) protocol. Applications connect to a data sharing group by specifying a DB2 location name. The group provides a single-system image to requesting applications. For more information about communicating with a data sharing group, see Chapter 4, “Communicating with data sharing groups,” on page 107.

**Administering a database**

Because the DB2 catalog is shared by all members of a data sharing group, data definition, authorization, and control are the same as for non–data sharing environments. Be sure that every object has a unique name, and be sure that the shared data resides on shared disks.

This section briefly describes some database administrative tasks and their special considerations for data sharing:

- “Planning for performance”
- “Planning for exit routines” on page 16
- “Authorizing users” on page 16
- “Loading and reorganizing data” on page 16

**Planning for performance**

When you create objects, the following options can affect data sharing performance:

- GBPCACHE
- MEMBER CLUSTER
- TRACKMOD

**GBPCACHE option:** Use the GBPCACHE option when you create or alter table spaces or indexes to specify what data, if any, should be cached in the group buffer pool. Valid values for this option are NONE, SYSTEM, CHANGED, and ALL. The default is CHANGED. See “Tuning group buffer pools” on page 236 for more information about choosing a valid value.

**MEMBER CLUSTER option:** Use the MEMBER CLUSTER option when you create table spaces to specify that DB2 locate data in the table space based on available space rather than clustering the data by the implicit or explicit clustering index.

This option can benefit applications when there are many inserts to the same table from multiple members. See “Reducing space map page contention” on page 245 for more information.

**TRACKMOD option:** Use the TRACKMOD option when you create or alter table spaces to specify whether you want DB2 to track modified pages in the space map pages of the table space or partition.

TRACKMOD NO can benefit applications when there are frequent updates from multiple members. Be aware that this option can degrade incremental image-copy performance; therefore, specify NO only if you never use incremental copies, or if you use DFSMS™ concurrent copies and LOGONLY recovery. In these cases,
choosing TRACKMOD NO can improve transaction performance. See “Reducing space map page contention” on page 245 for more information about these options.

Planning for exit routines
If you use exit routines, such as a field or validation procedure or the access control authorization routine, ensure that all members of the group use the same routines.

Recommendation: Place all exit routines in a program library that is shared by all members of the group.

Authorizing users
Use the same authorization mechanisms that are in place for non-data sharing DB2 subsystems to control access to shared DB2 data and to members. Because all members in the group share the same DB2 catalog, an authorization ID has the same granted privileges and authorities for every member of the group.

As suggested for non-data sharing DB2 subsystems, use a security system outside of DB2 (such as RACF® or its equivalent) to control which user IDs can access which members. RACF, for example, does not recognize a data sharing group as a single resource. Therefore, you must separately define DB2 resources to RACF for each member of the group, and connect all user IDs to a RACF group that permits access to all those resources. Or you can permit separate groups of user IDs to access different sets of resources. (In the latter case, however, you cannot move work freely among all members of the data sharing group.)

Each member of a data sharing group uses the same names for the connection and sign-on exit routines. As a good practice, all members of a group should share the same exit routines. Sharing avoids authorization anomalies such as:

• Primary authorization IDs that are treated differently by different members of the group
• Primary authorization IDs that are associated with different sets of secondary IDs by different members of the group

Loading and reorganizing data
You can load or reorganize data from any member of a data sharing group. For more information about LOAD and REORG utilities, see Part 2 of DB2 Utility Guide and Reference

Operating a data sharing group
This section describes some of the operational considerations for data sharing:

• “Issuing commands” on page 17
• “Recovering data” on page 17
• “Using coupling facilities effectively” on page 18
• “Stopping and starting DB2” on page 18
• “Maintaining a data sharing group” on page 18

Issuing commands
Parallel Sysplex technology lets you manage a data sharing group from a console that is attached to a single z/OS system or from consoles that are attached to multiple z/OS systems. Figure 13 on page 17 shows how commands are issued from a single z/OS system.
Using commands: Some commands manage group resources; others manage member resources. See the DB2 Command Reference for more information about specific commands.

Recovering data
DB2 recovers data from information that is contained in both the logs and the bootstrap data sets (BSDSs) of members. However, because updates can be logged on several different members, DB2 coordinates recovery by using the shared communications area (SCA) in a coupling facility. The SCA contains:

- Member names
- BSDS names
- Database exception status conditions about objects and members in the group
- Recovery information, such as log data set names and the list of indoubt XA transactions

The SCA is also used to coordinate startup.

The RECOVER utility: You can run the RECOVER utility from any member of a data sharing group. The process for data recovery is basically the same for a data sharing group as it is for non-data sharing DB2 subsystems. However, updates to a single table space can be the work of several different members. Therefore, to recover an object, DB2 must merge log records from the appropriate members, using a log record sequence number (LRSN). The LRSN is a value derived from the store clock timestamp and synchronized across the members of a data sharing group by the Sysplex Timer.

System-level point-in-time recovery: You can perform system-level point-in-time recovery by using the BACKUP SYSTEM and RESTORE SYSTEM online utilities.

BACKUP SYSTEM online utility: This utility provides fast, volume-level copies of DB2 databases and logs. It relies on DFSMSshm™ services in z/OS Version 1 Release 5 and higher. These services automatically keep track of which volumes
need to be copied. Using BACKUP SYSTEM is less disruptive than using SET LOG SUSPEND in copy procedures because the log write latch is not taken. An advantage for data sharing is that BACKUP SYSTEM has group-scope, whereas SET LOG SUSPEND has only member scope.

RESTORE SYSTEM online utility: This utility provides a way to recover a data sharing group to a specific point in time. RESTORE SYSTEM automatically handles any CREATE, DROP, and LOG NO events that might have occurred between the backup and the recovery point in time.

See “Recovering data” on page 160 for more information about recovery.

Using coupling facilities effectively
In addition to data objects, coupling facilities contain vital resources needed for data sharing.

Recommendation: Use more than one coupling facility to allow for structure duplexing and for automatic recovery in the event that a coupling facility fails. See “Coupling facility availability” on page 37 for more detailed suggestions.

Stopping and starting DB2
You can stop and start an individual member of a data sharing group while the other members continue to run. The startup process for each member is similar to that of non-data sharing DB2 subsystems.

DB2 uses a process called group restart in the rare event that critical resources in a coupling facility are lost and cannot be rebuilt. When this happens, all members of the group terminate abnormally. Group restart rebuilds the lost information from individual member logs. However, unlike data recovery, this information can be applied in any order. Because there is no need to merge log records, DB2 can perform many of the restart phases for individual members in parallel.

Recommendation: Use an automated procedure to restart failed members of the group.

Maintaining a data sharing group
To apply maintenance, you can make most changes on one member at a time, as shown in Table 1. If you must take DB2, IRLM, or z/OS offline for a change to take place and the outage is unacceptable to users, you can move those users onto another member. Some sites find it useful to define an extra member that they bring up and down as needed to take on work while maintenance is being applied to another member.

The recommended way of testing maintenance is to apply that maintenance to a test data sharing group before moving it onto the production data sharing group.

Table 1. Planned maintenance changes

<table>
<thead>
<tr>
<th>Type of change</th>
<th>Action required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early code</td>
<td>Bring down one z/OS system at a time and re-IPL.</td>
</tr>
<tr>
<td>DB2 code</td>
<td>Bring down and restart each member independently.</td>
</tr>
<tr>
<td>IRLM code</td>
<td>Bring down and restart each IRLM member independently.</td>
</tr>
<tr>
<td>Attachment code</td>
<td>Apply the change and restart the transaction manager or application.</td>
</tr>
</tbody>
</table>
Table 1. Planned maintenance changes (continued)

<table>
<thead>
<tr>
<th>Type of change</th>
<th>Action required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsystem parameters</td>
<td>For those that cannot be changed dynamically, update using the DB2 update process. Stop and restart the member to activate the updated parameter.</td>
</tr>
</tbody>
</table>

**Recommendation**: Consider specifying CASTOUT(NO) when you stop an individual member of a data sharing group for maintenance. This option speeds up shutdown because DB2 bypasses castout and associated cleanup processing in the group buffer pools.

Do not specify CASTOUT(NO) when you stop multiple members of a data sharing group and you need to maintain consistent data on disk. For example, if you stop all members to get a consistent copy of the databases on disk that you can send offsite, do not specify CASTOUT(NO) because some of the changed data could still reside in the group buffer pools after the members shut down. Your disk copy might not have all the most recent data.

**Tip**: Consider specifying a value of NODISCON for the IRLM procedure’s SCOPE option to allow IRLM to continue doing work while you apply maintenance to DB2. (If you edit the IRLM procedure using the DB2 installation process, this is analogous to specifying NO for parameter DISCONNECT IRLM on installation panel DSNTIPJ.) There are operational issues to be considered; see Part 2 of [DB2 Installation Guide](#) for more information about the ramifications of choosing this option.

**Applying maintenance to IRLM**: Each member of a data sharing group has its own IRLM. As with DB2, you must stop and restart each IRLM in the group to roll a change throughout the group.

IRLM has a function level to control changes that affect the entire group. The function level for a particular IRLM member indicates the latest function that IRLM is capable of running. The group function level is the lowest function level that exists among the IRLMs in the group. The group function level is the function level at which all IRLMs in the group must run, even if some IRLM members are capable of running at higher function levels.

**Recommendation**: Keep all members of the IRLM group at the same function level. This ensures that all IRLMs are running with the same maintenance that affects the entire group. (Maintenance that does not affect the group does not increment the function level.)

To see the IRLM function levels, use the MODIFY irlmproc,STATUS,ALLI command of z/OS. See “Determining the function level of the IRLM group” on page 93 for more information.

**Software and hardware requirements**

This section describes, at a high level, the software and hardware that are required to support DB2 data sharing. For detailed information about setting up a z/OS Parallel Sysplex, see [Parallel Sysplex Configuration Assistant](#).
Software

DB2 UDB for z/OS, Version 8 data sharing requires z/OS Version 1 Release 3 or later.

DB2 can run with any coupling facility level, but the most recent levels deliver the best function and performance.

- To be able to dynamically change structure sizes, structures must be allocated in a coupling facility at CFLEVEL=1 or higher, as described in “Changing structure sizes” on page 56.
- Coupling facility performance enhancements require CFLEVEL=2 or higher, as described in “Prefetch processing” on page 247 and “Locking optimizations” on page 219.
- Group buffer pool duplexing requires CFLEVEL=5 or higher. See “Duplexing structures” on page 42 for more information about group buffer pool duplexing.
- Group buffer pool checkpointing performs better when the group buffer pool is allocated in a CFLEVEL=5 coupling facility. Group buffer pool checkpointing is described in “Group buffer pool checkpoint” on page 253.

Check your coupling facilities to ensure that the appropriate service levels are installed. Having the wrong service levels installed can result in data corruption.

- If you have coupling facilities at CFLEVEL=7, you need service level 1.06 or above.
- If you have coupling facilities at CFLEVEL=8, you need service level 1.03 or above.

No specific service level requirements exist for CFLEVELs other than 7 and 8. Use the z/OS D CF command to display the service levels for IBM coupling facilities.

SCA and lock structure duplexing requires z/OS Version 1, Release 2 and CFLEVEL=10 or higher. It also requires DB2 Version 7 with APAR PQ45073 applied, and IRLM APAR PQ48823 applied.

Hardware

DB2 UDB for z/OS, Version 8 data sharing requires a zSeries® Parallel Sysplex, which includes:

- Central processor complexes (CPCs) that can attach to the coupling facility
- At least one coupling facility and the appropriate channels and channel cards
- At least one Sysplex Timer®
- Connection to shared disks

If you archive the DB2 log to tape, you might need a number of tape units greater than or equal to the number of members in the group. These tape units must be accessible and sharable by a member running a RECOVER utility.

Storage estimates

Installers must estimate the sizes of the various structures in the coupling facility. See “General information about coupling facility storage” on page 47 for more information.
Chapter 2. Planning for DB2 data sharing

To plan for the data sharing function of the licensed program DB2 UDB for z/OS, coordinate your efforts with system hardware and software groups. Complete these tasks before creating a data sharing group:

- “Planning for DB2 data sharing in a Parallel Sysplex”
- “Data sharing naming conventions” on page 27
- “Planning for availability” on page 35
- “Estimating storage” on page 46
- “Before you enable DB2 data sharing” on page 59
- “Application design planning” on page 68

The process of enabling data sharing is described in Chapter 3, “Installing and enabling DB2 data sharing,” on page 69.

If you already have a data sharing group on a release of DB2 previous to Version 8, read this chapter for new information, and see “Migrating an existing data sharing group to a new release” on page 91.

Planning for DB2 data sharing in a Parallel Sysplex

This section outlines planning for DB2 data sharing in a z/OS Parallel Sysplex, and describes the special connectivity needs for DB2 data sharing. For more information about specific hardware and software requirements for the z/OS Parallel Sysplex, see Parallel Sysplex Configuration Assistant.

# Parallel Sysplex components and requirements

This section describes the Parallel Sysplex and its relationship to DB2 data sharing. DB2 data sharing is dependent on the hardware and software components in the Parallel Sysplex.

Cross-system coupling facility component of z/OS

During startup, the members of a data sharing group join one cross-system coupling facility (XCF) group, and their associated internal resource lock managers (IRLMs) join another XCF group. The z/OS cross-system extended services (XES) also join an XCF group implicitly on behalf of the IRLM connection to the lock structure. To join a particular group, the data sharing group members and the IRLMs use the names you specify during DB2 installation.

DB2 uses the XCF for certain intersystem communications. Use both the coupling facility and channel-to-channel connections for XCF signalling. See z/OS MVS Setting Up a Sysplex for more information about configuring the XCF.

Sysplex timer

Install at least one Sysplex Timer in the Parallel Sysplex. For high availability, more than one Sysplex Timer is required. The Sysplex Timer synchronizes the timestamps of the z/Series processors for all members of the data sharing group. DB2 data sharing uses a value that is derived from the timestamp (as seen in the log) to recover data.
**Coupling facility**

Install and define at least one coupling facility to z/OS before enabling the DB2 data sharing function. For high availability, more than one coupling facility is required.

Data sharing member names from DSNZPARM are used to connect members to the coupling facility at DB2 startup. The first connector causes the list structure to be allocated in a coupling facility based on the preference list in the active CFRM policy.

**Coupling facility structures:** DB2 relies on areas of storage in the coupling facility called *structures*. Three types of structures exist: lock, list, and cache. Each structure type has a unique function. [Figure 14 on page 24](#) shows a sample configuration of the coupling facility structures used by DB2.

Members of a data sharing group use the following coupling facility structures to communicate and move data amongst themselves.

**Lock structure**

The lock structure protects shared DB2 resources (such as table spaces and pages) and enables concurrent access to those resources.

The system lock manager (SLM), a component of the z/OS cross-system extended services (XES), presents global lock information to the lock structure on behalf of each member’s IRLM.

The lock structure consists of two parts: a coupling facility lock list table (called the modified resource list), and a coupling facility lock hash table (called the lock table). The modified resource list records locks that protect changed data, thereby protecting the data in case of failure. The lock table contains the lock status information and the owning members of those locks, and is used to provide global lock serialization.

**List structure (SCA)**

The list structure contains the DB2 shared communications area (SCA). Each member uses the SCA to pass control information to the rest of the members in the group. The SCA contains all database exception status conditions and other information that is necessary for recovery of the group.

**Cache structures (DB2 group buffer pools)**

The cache structures are used as group buffer pools (GBPs), caching shared data pages for the members. You have the following options for caching data of interest to more than one member of a group:

- Cache all data (read-only and updated)
- Cache only data that is updated
- Cache only system control pages (specific to LOB table spaces)
- Cache no data; use the group buffer pool only for cross-invalidation

Group buffer pools use a cross-invalidation mechanism to maintain data consistency across the buffer pools of group members. Shared data pages are registered in a group buffer pool directory in each cache structure, thus enabling the coupling facility control program to cross-invalidate the copies of data pages that are held in individual member buffer pools.

Cross-invalidation takes place when a member’s own buffer pool does not contain the latest version of the data. In this case, the member must reread the pages from either the group buffer pool or disk, if that member needs to reference the pages again.
One group buffer pool exists for all member buffer pools of the same name. For example, each member must have a buffer pool 0 (BP0) that contains the catalog and directory table spaces. A group buffer pool 0 (GBP0) must exist on a coupling facility for that data sharing group.

Similarly, if a member creates table space X and associates it with buffer pool 1 (BP1), X is associated with BP1 for every member because there is only one definition of X in the catalog for the entire group. To share the data in X, you must define the cache structure, group buffer pool 1 (GBP1). If you do not define the group buffer pool, a single member can update X or more than one member can read X, but there can be no inter-DB2 read/write activity for X.

**Recommendation:** For data that is private to each member, such as work files or user data that only one member reads, define that data to a buffer pool for non-shared page sets. For example, assume that you want to associate all non-shared page sets with buffer pool 6 (BP6). If you want only member DB2A to access a non-shared table space Y, then define Y (and any indexes) to BP6. Define BP6 with a size of 0 (zero) and then you do not need to define the coupling facility structure for group buffer pool 6. By moving private data to buffer pools that are separate from buffer pools used by shared data, you can more easily monitor, and provide for more predictable performance of, private data.

**Defining coupling facility structures:** Before you enable DB2 data sharing, you must define coupling facility structures. Use the z/OS coupling facility resource management (CFRM) policies to define these structures to the Parallel Sysplex. A CFRM policy determines how and where the structure resources are allocated.

You must define one lock structure, one list structure, and at least four cache structures:

- Group buffer pool 0
- Group buffer pool 8K0
- Group buffer pool 16K0
- Group buffer pool 32K

The lock and list structures do not need to be in the same coupling facility. Individual structures cannot span coupling facilities.
**Recommendation:** Define the availability characteristics of the coupling facility structures for lost connectivity failures, which includes a total failure of the coupling facility. Define a Sysplex Failure Management (SFM) policy, as described in "Rebuilding structures when connectivity is lost" on page 40.

See z/OS MVS Setting Up a Sysplex for information about how to create CFRM and SFM policies.

A sample CFRM policy is shown in Figure 15 on page 25.

*Figure 14. Coupling facility structures used by DB2. This is a sample configuration. The lock structure and list (SCA) structure do not need to be in the same coupling facility.*
For DB2, you must know the following characteristics of DB2 coupling facility structures before you create the policy definitions:

- Initial size and maximum size of the structures
  
  See “General information about coupling facility storage” on page 47.
  
  The structures can be dynamically resized from INITSIZE up to the value in SIZE. See “Changing structure sizes” on page 56 for more information.

- Structure names
  
  See “Coupling facility structure names” on page 31.

- Availability characteristics
  
  You must know the preference list (PRELIST) for rebuilding or reallocating a structure, if the coupling facility fails. See “Coupling facility availability” on page 37 for more information.

Figure 15. Sample CFRM policy

For DB2, you must know the following characteristics of DB2 coupling facility structures before you create the policy definitions:

- Initial size and maximum size of the structures
  
  See “General information about coupling facility storage” on page 47.
  
  The structures can be dynamically resized from INITSIZE up to the value in SIZE. See “Changing structure sizes” on page 56 for more information.

- Structure names
  
  See “Coupling facility structure names” on page 31.

- Availability characteristics
  
  You must know the preference list (PRELIST) for rebuilding or reallocating a structure, if the coupling facility fails. See “Coupling facility availability” on page 37 for more information.
Authorize DB2 to access the structures: Optionally, you can set up a facility class profile to limit access to the structures in the coupling facility. If you do this, ensure that DB2 does have access by ensuring that the IDs associated with the DB2 address spaces have alter access authority to the coupling facility structures through RESOURCE(IXLSTR.structure_name) in SAF class CLASS(FACILITY).

If you do not create a facility class profile, the default allows any authorized user or program (supervisor state and program key mask allowing key 0-7) to issue coupling facility macros for the structure.

Common z/OS libraries
As stated in “Naming recommendations” on page 32, DB2 supports a configuration with a SYS1.PARMLIB and SYS1.PROCLIB that is shared by all z/OS systems in the Parallel Sysplex. This configuration lets you add and modify systems more easily.

If you intend to have many members in the Parallel Sysplex, each DB2 and IRLM that you define to the z/OS system in the IEFSSNx.xx parmlib member requires a z/OS system linkage index (LX). The default number of these indexes that z/OS reserves is 165. If you place all of your DB2 and IRLM subsystem definitions in a single IEFSSNx.xx member, you might need more than 165 LXs to start the members.

If you need more than 165 LXs, use the NSYSXL option on the z/OS IEASYSxx parmlib member to increase this number. See z/OS MVS Initialization and Tuning Guide for more information.

Connectivity requirements
DB2 data sharing requires that all DB2-related resources reside on shared disks. The DB2 catalog and directory and any user data that is shared must be on shared disks. The integrated catalog for DB2 data sets must also be on shared disks.

Also, all the members’ logs and bootstrap data sets (BSDSs) must be on shared disks for recovery purposes. A member performing recovery must have access to the logs of other members in the group.

Recommendation: Place work files on shared disks for the following reasons:
- For queries processed using Sysplex query parallelism, the placement of work files on shared disks is a requirement. Each assisting member writes to its own work file, and the coordinator can read the results from the assistants’ work files.
- A member stays connected to its work file even if you need to restart the member on another processor.
- You can create or drop a work file table space from any other member in the data sharing group.

Make sure that you have physical connectivity by checking the following connections:
- Verify that one user-integrated catalog facility exists for cataloging the data sets of a data sharing group, and that you can access this catalog from each z/OS system in the Parallel Sysplex.
• Verify connectivity to the following entries from each system on which a member resides:
  – A set of DB2 target libraries
  – A single DB2 catalog
  – A single DB2 directory
  – All databases that are shared
  – All log data sets
  – All BSDS data sets
  – All coupling facilities used by the data sharing group
  – User integrated catalog facility catalogs for shared databases

# Data sharing naming conventions

Carefully consider the naming convention you will use to name the various parts of the data sharing system. Assign names to both IRLM and data sharing groups, and to members within a group. One recommendation is to make names and prefixes unique within the Parallel Sysplex. Although this uniqueness is not required for all names, it helps you avoid problems with identifying and moving entities among z/OS systems in the Parallel Sysplex.

This section describes the names for which you must choose values. Other names are generated during DB2 installation. A complete list of names is in “DB2 and IRLM names,” on page 275. “Naming recommendations” on page 32 describes a suggested naming convention. This naming convention is used for the names in this section. If you want to change the name of an existing DB2 subsystem to conform to your naming convention, see “Renaming a member” on page 78.

Data sharing group names

The following names are shared by all members of the data sharing group.

DB2 group name

The name that encompasses the entire data sharing group. The coupling facility structure names are based on this name.

The DB2 group name must be unique within the Parallel Sysplex. If you use this name as a basis for the location name, the DB2 group name must be unique within the network.

This name can be up to eight characters long, it must begin with an alphabetic character, and it can consist of the characters A-Z, 0-9, $, #, and @. An example of a DB2 group name is DSNDB0A.

Restrictions: To avoid names that IBM uses for its XCF groups, do not begin DB2 group names with the letters A-I unless the first three characters are DSN. Do not use the string SYS as the first three characters, and do not use the string UNDESIG as the group name.

Important: Never reuse a DB2 group name, even if a data sharing group that previously used the name no longer exists. Some data sharing information, such as the DB2 group name, is retained in the Parallel Sysplex couple data set (CD5). To determine what DB2 group names exist, execute the z/OS command DISPLAY XCF,GROUP.

ICF catalog alias

The name of the Integrated Catalog Facility (ICF) catalog alias that you must place in the z/OS master catalog. This name can be up
to 8 characters long, and it should be the same as the DB2 group name. An example of an ICF catalog alias name is DSNDDB0A.

**Group attachment name**
The name that is used by the TSO/batch attachment, the call attachment facility (CAF), DL/I batch, utilities, and the Resource Recovery Services attachment facility (RRSAF) as a “generic” attachment name. This name can be up to 4 characters long. An example of a group attachment name is DB0A.

For more information about using the group attachment name, see “Using the group attachment name” on page 148.

**SQL port number**
The port number that is used by the distributed data facility (DDF) to accept incoming SQL connection requests, if the data sharing group uses TCP/IP. 446 is the recommended DRDA® port. See "DRDA access through TCP/IP” page 129 for more information about using TCP/IP.

**Location name**
The name that is used when the data sharing group processes distributed requests. The group is treated as a single location by remote requesters. This name can be up to 16 characters long.

**Generic LU name**
This name lets remote requesters configure their systems to treat the data sharing group as a single LU. This name can be up to eight characters long. See Chapter 4, “Communicating with data sharing groups,” on page 107 for more information about a group’s generic LU name.

**Parallel Sysplex domain name**
This name lets you take advantage of workload balancing for TCP/IP connections. If your site uses a domain name server (DNS), you must register this name with the server. See “Registering names in the domain name server” on page 119 for more information about the Parallel Sysplex domain name.

**Member names**
The following names must be unique within the data sharing group or, in certain cases, the z/OS Parallel Sysplex.

**Member name**
The name of an individual member of a data sharing group. A member name can be up to eight characters long, and it can consist of the characters A-Z, 0-9, $, #, and @. DB2 uses this name to form its z/OS cross-system coupling facility (XCF) member name. An example of a member name is DB1A.

**DB2 subsystem name**
The name of a DB2 subsystem that is a member of a data sharing group. This name is used by all the attachment interfaces. A DB2 subsystem name can be up to four characters long, and it must be unique within the Parallel Sysplex. The member name and the DB2 subsystem name should be the same. An example of a DB2 subsystem name is DB1A.

**LU name**
The network name of an individual member of a data sharing group. A logical unit (LU) name must be unique both within the
group and within the network. See Part 3 of **DB2 Installation Guide** for more information about choosing LU names for members of data sharing groups.

**Member domain name**

This name lets DB2 handle indoubt thread resolution for TCP/IP connections. You must register each member’s domain name with the domain name server (DNS). See “Registering names in the domain name server” on page 119 for more information about the member domain name.

**Resynchronization port**

The TCP/IP port number that is used by the distributed data facility (DDF) to accept incoming DRDA two-phase commit resynchronization requests, if the data sharing group uses TCP/IP. Each member of a data sharing group listens on a unique resynchronization port. See “Group access” on page 110 for more information.

**Command prefix**

The prefix that directs commands entered at a z/OS console to a particular member of a data sharing group. The command prefix can be up to 8 characters long. The default prefix is the concatenation of the hyphen character (-) with the DB2 subsystem name. See Part 2 of **DB2 Installation Guide** for more information about valid characters for a command prefix.

As described in “Registering command prefixes and member group attachment name” on page 61, this string is specified as a parameter on the IEFSSNxx subsystem definition. An example of a command prefix is -DB1A. You can have blanks between the command prefix and the command.

Do not assign a command prefix that is used by another DB2 subsystem or that can be interpreted as belonging to more than one DB2 subsystem or z/OS application. Specifically, do not specify a multiple-character command prefix that is a subset of or a superset of another command prefix starting from the first character. For example, it is invalid to assign a hyphen (-) to one DB2 subsystem and ‘-DB2A’ to another DB2 subsystem. Similarly, it is invalid to assign ‘?DB2’ to one DB2 subsystem and ‘?DB2A’ to another DB2 subsystem. It is valid, for example, to assign ‘-DB2A’ and ‘-DB3A’ to different DB2 subsystems.

**Work file database**

The name of the work file database that is associated with a member of the data sharing group. Each member has its own work file database. In a non-data sharing environment, a DB2 subsystem’s work file database is called DSNDB07. In a data sharing environment, although one member of the group can have a work file database with the name DSNDB07, you might want to create one with a more meaningful name, such as WRKDB1A, for member DB1A. This name can be up to 8 characters long.

**Restriction:** You cannot specify a name that begins with DSNDB unless the name is DSNDB07.

**Load module for subsystem parameters**

The name of the load module that contains a member’s subsystem parameters. Each member has its own subsystem parameters. This
Choosing names for member data sets: When choosing names for member data sets, remember that data set names beginning with *membname* must have a master catalog alias to point to the catalog where the data sets are cataloged. The DB2 installation process does not create this catalog alias. One way to handle this is to begin member data set names with *catalias* and a member-related qualifier. For example, member data set names could take the form *catalias.membname.xxxxx*. This format eliminates the need to have a master catalog alias for *membname*.

Member BSDS names
The names of bootstrap data sets (BSDS) belonging to a member of a data sharing group. These names can be up to 33 characters long. Example BSDS names are DSNDB0A.DB1A.BSDS01 and DSNDB0A.DB1A.BSDS02.

Active log data set prefixes
The prefixes of active log data sets belonging to a member of a data sharing group. These prefixes can be up to 30 characters long. Example active log data set prefixes are DSNDB0A.DB1A.LOGCOPY1 and DSNDB0A.DB1A.LOGCOPY2.

Archive log data set prefixes
The prefixes of archive log data sets belonging to a member of a data sharing group. These prefixes can be up to 35 characters long unless you want the data sets time stamped. If they are time stamped, these prefixes can be up to only 19 characters long. Use the TIMESTAMP ARCHIVES parameter of installation panel DSNTPRH to specify whether you want the date and time qualifiers to be added to the archive log data set prefix.

Example archive log data set prefixes are DSNDB0A.DB1A.ARC1 and DSNDB0A.DB1A.ARC2.

IRLM names
Each member of a data sharing group has its own IRLM. The IRLM group name, subsystem name, and member ID are parameters on the IRLM startup procedure. This means that every IRLM in the group must have a separate IRLM procedure. Figure 16 shows the relationship between DB2 group names and IRLM group names.

![Figure 16. Relationship between DB2 and IRLM group names](image)

You must choose the following IRLM names before installing DB2:

**Group name** The name that encompasses the entire IRLM group. This name can
be up to 8 characters long and can consist of the characters A-Z, 0-9, $, #, and @. The group name must begin with an alphabetic character.

**Restrictions:** To avoid names that IBM uses for its XCF groups, do not begin with the letters A-I unless the first three characters are DXR. Do not use the string SYS as the first three characters, and do not use the string UNDESIG as your group name.

The IRLM group name is a parameter (IRLMGRP=) on each member's IRLM procedure. An example IRLM group name is DXRDB0A, a naming convention that you can use to associate member DB0A with DXRDB0A.

**Subsystem name**
Each IRLM must have a subsystem name that can be up to 4 characters long. This name is a parameter on the IRLM procedure.

A sample subsystem name is DJ1A. The “1A” characters indicate that this IRLM is paired with the member DB1A.

**IRLM procedure name**
Each member of a data sharing group knows its IRLM by the procedure and subsystem names saved in that member's installation parameter load module. The IRLM procedure name can be up to 8 characters long.

**Recommendation:** Use the member subsystem name followed by “IRLM.” An example IRLM procedure name is DB1AIRLM.

**IRLM member ID**
This ID uniquely names an IRLM within a group. It is a number between 1 and 255 (inclusive). See the description of the IRLMID parameter of the START irlmproc command in Part 3 of [DB2 Command Reference](#) for more information about this value.

## Coupling facility structure names

Names for coupling facility structures must conform to a strict naming convention based on the DB2 group name. Sample names are shown in [Table 2](#).

<table>
<thead>
<tr>
<th>Structure type</th>
<th>Example name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache structure (group buffer pools)</td>
<td>DSNDB0A_GBP0</td>
</tr>
<tr>
<td>Lock structure</td>
<td>DSNDB0A_LOCK1</td>
</tr>
<tr>
<td>List structure (shared communications area)</td>
<td>DSNDB0A_SCA</td>
</tr>
</tbody>
</table>

**Lock structure name**
Use a name of the following format on the CFRM policy to define the lock structure to the coupling facility:

```
groupname_LOCK1
```

**Shared communications area**
Use a name of the following format on the CFRM policy to define the SCA to the coupling facility:

```
groupname_SCA
```

---

Chapter 2. Planning for DB2 data sharing  31
Group buffer pool names

Use a name of the following format on the CFM policy to define the group buffer pool to the coupling facility:

\textit{groupname\_GBPxxxx}

Where GBPxxxx is the name of the group buffer pool; the following restrictions apply:

- 4 KB group buffer pools are named GBP0, GBP1, ..., GBP49
- 8 KB group buffer pools are named GBP8K0, GBP8K1, ..., GBP8K9
- 16 KB group buffer pools are named GBP16K0, GBP16K1, ..., GBP16K9
- 32 KB group buffer pools are named GBP32K, GBP32K1, ..., GBP32K9

\textbf{Requirement:} You must define GBP0, GBP8K0, GBP16K0, and GBP32K. You must also define any other group buffer pools that you need for your particular data sharing setup.

When DB2 duplexes a group buffer pool structure, the same structure name is used for both the primary and secondary structures. A duplexed structure requires a single CFM policy structure definition, with one structure name.

Naming recommendations

You control the names you assign to entities during the DB2 installation process. After installation, you cannot change some names, such as the group name and the member names. Because you must name a large number of items, and you might have to add new members or move existing members to different systems in the future, managing all these items is easier if you choose and maintain a consistent naming convention.

If you are enabling the originating member of the group from an existing DB2 subsystem, you can build a naming scheme around existing names to reduce the disruption to existing applications. However, before enabling data sharing, you might want to change some names to lay the foundation for a solid naming convention.

Another name to choose carefully is the catalog alias for the group. It is very difficult to change that name. The procedure to do this for a single system is documented in Part 2 (Volume 1) of \textit{DB2 Administration Guide}. To change the catalog alias for the group, you must bring the entire group down and perform the single-system procedure for every member of the group.

Configuration assumptions

DB2 data sharing naming recommendations and installation process support assume the following z/OS Parallel Sysplex configuration:

- One SYS1.PARMLIB shared by all z/OS systems
- One SYS1.PROCLIB shared by all z/OS systems
- One integrated catalog facility master catalog shared by all z/OS systems

DB2 data sharing does not require that the Parallel Sysplex be configured in this manner. However, the following DB2 naming recommendations support such a configuration, and the DB2 installation process assumes such a configuration.

If the z/OS Parallel Sysplex is configured differently, you must customize the installation process. For example, if you use different SYS1.PARMLIBs, make sure that the DB2 data sets are in the APF list in each PARMLIB. If you use different
PROCLIBs, modify the JCL to point to the correct libraries during installation. And, if you are using more than one integrated catalog facility (IFI) master catalog, put the DB2 catalog alias in each master catalog.

**Naming convention suggestions**
Consider the following suggestions when naming various DB2 entities:

- **Name subsystems first.**
  Subsystem names are limited to four characters, making them the shortest names. After you select subsystem names, use them as the basis for creating other names.
  
  Four-character names are also needed for the following entities:
  - DB2 group name; one per data sharing group
  - Member name; one for each DB2 subsystem in the group
  - IRLM subsystem name; one for each member of the group
  
  One possible naming convention is the one suggested in *z/OS Parallel Sysplex Application Migration*, which takes a Parallel Sysplex-wide approach to naming. That convention assigns subsystem names of the form ‘ctmg’ where:

  - **c** Denotes a particular collection of logically related applications or subsystems.
  - **t** Denotes a type of resource, such as “B” for DB2 or “J” for IRLM.
  - **m** Denotes a particular member within a data sharing group or an IRLM group. This identifier also associates a subsystem with its z/OS system for recovery when you use automatic restart.
  - **g** Denotes a particular DB2 group.

  This book uses A as the group identifier. Therefore, the naming scheme has a subsystem name of DB1A for the first member of the data sharing group. The second member is DB2A, and so on. Table 3 shows how the member identifier relates to a particular z/OS system and how the group identifier associates members across a Parallel Sysplex system. In each column are the names associated with a particular z/OS recovery group (z/OS2, for example). In each row are the names associated with the primary components of the data sharing group, such as DB2 subsystems and IRLM subsystems.

<table>
<thead>
<tr>
<th>z/OS1</th>
<th>z/OS2</th>
<th>z/OS3</th>
<th>z/OS4</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA11 (CICS)</td>
<td>DA21 (CICS)</td>
<td>DA31 (CICS)</td>
<td>DA41 (CICS)</td>
</tr>
<tr>
<td>DB1A (DB2)</td>
<td>DB2A (DB2)</td>
<td>DB3A (DB2)</td>
<td>DB4A (DB2)</td>
</tr>
<tr>
<td>DJ1A (IRLM)</td>
<td>DJ2A (IRLM)</td>
<td>DJ3A (IRLM)</td>
<td>DJ4A (IRLM)</td>
</tr>
<tr>
<td>DD11 (DBCTL)</td>
<td>DD21 (DBCTL)</td>
<td>DD31 (DBCTL)</td>
<td>DD41 (DBCTL)</td>
</tr>
</tbody>
</table>

- If you enable an existing DB2 subsystem to take advantage of data sharing, and existing applications already access that DB2 subsystem, consider using the name of the existing DB2 subsystem as the group attachment name. This lets existing applications continue to use the data sharing member; it does not require any changes to those applications, in terms of specifying a new subsystem name.
- Use the same name for the DB2 group name, the DB2 location name, and the DB2 integrated catalog alias.
A data sharing group uses the same catalog alias name. This catalog alias is used as the high-level qualifier for the DB2 directory (DSNDB01), catalog (DSNDB06), default database (DSNDB04), and work file database VSAM data sets.

- Use the same name for a member’s member name and DB2 subsystem name.
- Accept the default prefix for a member’s command prefix.

**Naming example for DB2 data sharing**

Table 4 shows an example of the names chosen for a 12-member data sharing group. The example uses the subsystem naming convention suggested in the section “Suggested Naming Conventions.” A zero in place of the member identifier is used to identify a group-wide resource, such as the group name or the catalog alias (DSNDB0A). The letter ‘A’ indicates that this naming convention is for data sharing group A. You can establish similar naming conventions for other data sharing groups, for example, group DSNDB0B. In Table 4, the '#' character is used to denote a character from the set {1-9,A,B,C}. These characters are used to denote a particular member’s name.

**Table 4. DB2 data sharing naming example**

<table>
<thead>
<tr>
<th>Entity</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 group name</td>
<td>DSNDB0A</td>
</tr>
<tr>
<td>Catalog alias</td>
<td>DSNDB0A</td>
</tr>
<tr>
<td>Group attachment name</td>
<td>DB0A</td>
</tr>
<tr>
<td>Member name</td>
<td>DB#A</td>
</tr>
<tr>
<td>DB2 subsystem name</td>
<td>DB#A</td>
</tr>
<tr>
<td>Command prefix</td>
<td>-DB#A</td>
</tr>
<tr>
<td>Work file database</td>
<td>WRKDB#A</td>
</tr>
<tr>
<td>Load module for subsystem parameters</td>
<td>DSNZP0#A</td>
</tr>
<tr>
<td>Member BSDS names</td>
<td>DSNDB0A.DB#A.BSDS01</td>
</tr>
<tr>
<td></td>
<td>DSNDB0A.DB#A.BSDS02</td>
</tr>
<tr>
<td>Active log data set prefixes</td>
<td>DSNDB0A.DB#A.LOGCOPY1</td>
</tr>
<tr>
<td></td>
<td>DSNDB0A.DB#A.LOGCOPY2</td>
</tr>
<tr>
<td>Archive log data set prefixes</td>
<td>DSNDB0A.DB#A.ARC1</td>
</tr>
<tr>
<td></td>
<td>DSNDB0A.DB#A.ARC2</td>
</tr>
<tr>
<td>IRLM group name</td>
<td>DXRDDB0A</td>
</tr>
<tr>
<td>IRLM subsystem name</td>
<td>DJ#A</td>
</tr>
<tr>
<td>IRLM procedure name</td>
<td>DB#AIRLM</td>
</tr>
<tr>
<td>IRLM member ID</td>
<td>Number 1-12 corresponding to #</td>
</tr>
</tbody>
</table>

**Distributed naming convention:** The example in Table 4 does not include names for distributed processing. Those naming conventions will probably be part of a much broader convention. See your network administrator for more information about choosing names for distributed access.
Planning for availability

When planning your data sharing configuration for the highest availability and best network configuration, the physical protection of the coupling facility and the structures within the coupling facility, as well as the highest network connectivity, are the primary concerns. The SCA and lock structure are both necessary for the group to function.

For the highest availability:

1. Duplex the group buffer pool to allow DB2 to switch to the secondary structure if the primary structure fails.

   Although the loss of a group buffer pool does not require a group restart, availability for users and important applications requires that data in a group buffer pool be available as quickly as possible after failure. Group buffer pools have several availability options, depending on the type of failure that occurs. Duplexing group buffer pool structures assures minimal impact on performance. Duplexing can also help you to avoid hours of recovery time, which are often required to recover simplexed group buffer pool structures. If a simplexed group buffer pool structure fails, the group buffer pool can be recovered automatically from data in the DB2 logs. If members lose connectivity to the group buffer pool, the group buffer pool can be rebuilt in another coupling facility to which the members can connect. Recovering data from the DB2 logs and rebuilding structures in another coupling facility can be very time-consuming.

2. Configure a simplexed SCA and lock structure in failure-isolated coupling facilities. (When duplexed, the SCA and lock structure do not need to be in failure-isolated coupling facilities.)

   Duplexing the SCA and lock structure offers marginally faster recovery times compared to dynamically rebuilding a simplexed SCA and lock structure. Duplexing also enables you to use internal coupling facilities (ICFs), which provide cost savings over stand-alone coupling facilities without compromising availability. However, although you can duplex the SCA and lock structure, before doing so, consider that:
   
   • Duplexing the SCA and lock structure does not add significant availability, because the structures can be rebuilt very quickly from in-memory data.
   
   • Duplexing the SCA and lock structure can cause a significant impact on performance. In most cases, the overhead that is incurred by duplexing these structures outweighs the availability benefits of duplexing.

3. Define a dynamic virtual IP address (VIPA) for the DB2 group and a dynamic VIPA for each DB2 member in the group. Dynamic VIPA addressing gives you the ability to assign a specific virtual IP address to a data sharing group and to each member of the group. This address is independent of any specific TCP/IP stack within the Parallel Sysplex. Even if a member is moved to another z/OS system, as in the case of a failure or maintenance, the member remains accessible and retains the same virtual IP address.

   • To achieve the highest level of application availability at the database level, route all connections through a DB2 Connect Server with Sysplex support enabled. To route all connections through a DB2 Connect Server with Sysplex support enabled, you must first configure the DB2 group and each DB2 member to use dynamic VIPA. The DB2 Connect server then manages the connections to the group using the dynamic VIPAs.

   • To achieve the highest level of Web application availability using JTA and JTS distributed transactions against a data sharing group, use the DB2 Universal

Chapter 2. Planning for DB2 data sharing 35
JDBC Driver and WebSphere Application Server as the transaction manager in a distributed transaction system. The DB2 Universal JDBC Driver and WebSphere Application Server use dynamic VIPA to coordinate distributed transactions against the DB2 group.

The following sections describe these topics:
- “Automatic restart of z/OS” on page 37
- “Coupling facility availability” on page 37
- “Duplexing structures” on page 42
- “DB2 resource availability considerations” on page 44
- “Planning for WebSphere coordinated XA transactions” on page 45

Automatic restart of z/OS
The purpose of automatic restart is to reduce the time that a particular system is down. When DB2 or IRLM stops abnormally, the surviving z/OS systems analyze the situation to determine whether z/OS failed too, and where DB2 or IRLM should be restarted. If z/OS did not fail, then DB2 or IRLM are restarted on the same z/OS image. If z/OS failed, then DB2 or IRLM are restarted on another z/OS image.

Automatic Restart Manager (ARM) is the component of z/OS that manages automatic restarts. To use ARM, you need to set up an ARM policy, as described in “Creating an automatic restart policy.”

Advantage of automatic restart
When DB2 and IRLM restart automatically, locks held by failed members are released quickly. Applications that are running on other members can then access data for which the failed member is holding incompatible locks. Consider using automatic restart in conjunction with the RETAINED LOCK TIMEOUT option of installation panel DSNTIPI described in Table 14 on page 73.

You must install DB2 with a command prefix scope of “started” to take advantage of automatic restart. See “Registering command prefixes and member group attachment name” on page 61 for more information.

Using an automatic restart policy
You control how automatic restart works by using z/OS automatic restart policies. When the automatic restart function is active, the default action (for both sharing and non-sharing DB2 subsystems and IRLMs alike) is to restart the subsystems when they fail. If this default action is not what you want, you must create a policy that defines the action you want taken.

Creating an automatic restart policy
If the default action of restarting DB2 is the action you want, you do not need to create a policy. However, if you want to change the default behavior, you need to know the automatic restart ELEMENT name. In a data sharing group, the DB2 element name is the DB2 group name concatenated with the member name (such as DSNDDB0ADB1A).

For IRLM, the element name is the IRLM group name concatenated with the IRLM subsystem name and three-character member ID (such as DXRDB0ADJ1A001).

You can also specify a pattern-matching character (such as DSNDDB0A*) if you want to use a single policy statement for all members in the group.
DB2 startup can be a little faster when ARM restarts IRLM because this activity is done in parallel. DB2 does not have to start IRLM and wait.

To specify that DB2 or IRLM is not to be restarted after a failure, include RESTART_ATTEMPTS(0) in the policy for that DB2 or IRLM element. For IRLM, you can also use the MODIFY irlmproc,ABEND command, as shown below, to stop IRLM and unregister it from ARM when it comes down. Unregistering prevents IRLM from automatically restarting after you bring it down.

MODIFY irlmproc,ABEND,NODUMP

However, if a member’s AUTO START option of installation panel DSNTP1 has a value of YES, and if z/OS restarts DB2 automatically, DB2 restarts IRLM, too.

**Restart light:** Restart light enables DB2 to restart with a minimal storage footprint to quickly release retained locks and then terminate normally. It is not recommended for a restart in place, but it is recommended for a cross-system restart in the event of a failed z/OS system. It is primarily intended to restart DB2 temporarily on another z/OS system that does not have the capacity to sustain a DB2 and IRLM pair. ARM will not restart this member again when it comes down after performing a restart light.

**Policy requirements for restart light:** To have DB2 restarted in a light mode (restart light), you must have an ARM policy for the DB2 group that specifies LIGHT(YES) within the RESTART_METHOD(SYSTEM) keyword for the DB2 element name. For example:

RESTART_METHOD(SYSTEM,STC,'cmdprfx STA DB2,LIGHT(YES)')

### Coupling facility availability

For high availability, you should have at least two coupling facilities, and at least one of them should be non-volatile. With multiple coupling facilities, you can specify that structures be allocated in the secondary coupling facility, if the primary coupling facility is damaged. You can also consider duplexing SCA, lock, and group buffer pool structures. With duplexing, a secondary structure is always on standby in another coupling facility. This secondary structure is ready to take over if the primary structure fails or if a connectivity failure occurs. If you have three or more coupling facilities, you can even maintain duplexing while performing maintenance on one of the coupling facilities.

Duplexing the group buffer pool structures is a good idea because the performance overhead of duplexed group buffer pools are negligible in most cases, and the availability benefits are very high. Duplexing the SCA and lock structure is not as important for high availability because these structures can be rebuilt dynamically on an alternate coupling facility, if the coupling facility that contains the SCA and lock structure fails.

If the SCA and lock structure are not duplexed, the coupling facility that contains these structures should be failure isolated for the highest availability. A failure-isolated CF resides in a central processor complex (CPC) that does not also contain a data sharing member that is connected to structures in that coupling facility. If the SCA and lock structure reside in a non-failure-isolated coupling facility (a coupling facility that contains the SCA and lock structure and resides in a CPC that also contains a member of that data sharing group), the CPC becomes a single point of failure. If the CPC fails, the entire data sharing group comes down. Duplexing the SCA and lock structure, or keeping the SCA and lock structure in a failure-isolated coupling facility, avoids this single point of failure.
Requirement: SCA and lock structure duplexing requires z/OS Version 1 Release 2 or higher.

For more information about duplexing structures, see “Duplexing structures” on page 42.

See Parallel Sysplex Configuration Assistant for more information about configuring the coupling facility for high availability.

Recommendations for placement of coupling facilities
Physically separate the coupling facilities from each other and from the rest of the z/OS images that use those coupling facilities. If you separate the SCA and lock structures from the systems that use them, you can minimize the chances of performing a lengthy group restart after a lengthy outage.

Quick recovery of the group buffer pools uses information in the lock structure and SCA to determine which databases must be recovered. This is known as damage assessment. Consider putting the lock structure and SCA in a coupling facility that does not contain important cache structures (such as group buffer pool 0). You are less likely to lose the SCA, lock structure, and the group buffer pool at the same time if you carefully separate these structures by placing them in different coupling facilities.

If you lose the lock structure or SCA at the same time as one or more group buffer pools, DB2 waits until the lock structure and SCA are rebuilt before doing damage assessment.

Preparing for coupling facility failures
The major types of coupling facility failures are structure failures, connectivity failures, and channel failures:

- A structure failure occurs when a group buffer pool, lock, or SCA structure no longer operates.
- A connectivity failure is a total failure of the coupling facility, such as a power failure to the coupling facility or some problem with the coupling facility control code.
- A channel failure occurs when a channel connecting a CPC to a coupling facility no longer operates.

Recommendation: To assure high availability when preparing for structure and connectivity failures, configure the group buffer pool structures in duplex mode and the SCA and lock structures in simplex mode. To prepare for channel failures, consider using dual channels between each CPC and a coupling facility.

Preparing for structure failures
If structure failures occur, DB2 can recover resources in the structures. This section outlines the recovery scenarios for duplexed and simplesxed structures.

Duplexed group buffer pool structure failures: If a duplexed group buffer pool structure fails, the failing structure is deallocated, and processing continues with the running structure. This recovery is usually fast and unnoticeable.

If both group buffer pool structures fail, recovery is performed as described in “Simplexed group buffer pool structure failures” on page 39. Use checkpointing, as described in “Tuning the group buffer pool checkpoint interval” on page 253, in case both structures fail.
Simplexed group buffer pool structure failures: Simplexed group buffer pools can be recovered from the log when they fail. Recovery from the log can occur manually, as the result of a START DATABASE command, or it can occur automatically because the group buffer pool is defined with the AUTOREC(YES) option. In either case, to reduce the time that is needed for group buffer pool recovery, use the ALTER GROUPBUFFERPOOL command to make group buffer pool checkpoints more frequent. However, weigh the benefit of faster recovery with the processing resources that are used for the checkpointing. (You can reduce checkpoint costs considerably when z/OS is at the appropriate level of maintenance. See “Tuning the group buffer pool checkpoint interval” on page 253 for more information.)

Also consider using the DB2 fast log apply process to speed up recovery. To enable the fast log apply process, indicate how much ssmnDBM1 storage can be used for the log apply function on the LOG APPLY STORAGE parameter of installation panel DSNTIPL.

Ensure that you specify one or more alternate coupling facilities in the CFRM preference list for the group buffer pools because a group buffer pool can be allocated in an alternative coupling facility when a new connection is made to it. See “Problem: group buffer pool structure failure (no duplexing)” on page 176 for more information about group buffer pool structure failures.

More about automatic recovery: Enable automatic recovery with the AUTOREC option of the ALTER GROUPBUFFERPOOL command. Automatic recovery is faster than manual recovery because DB2 can optimize internal processing of the recovery. For automatic recovery to be initiated for a group buffer pool, all of the following conditions must be true:

- AUTOREC (YES) is specified for the group buffer pool.
- At least one member is actively connected at the time that the group buffer pool failed. This member must have successfully completed damage assessment.

DB2 never initiates automatic recovery during restart.

For a duplexed group buffer pool, DB2 can use automatic recovery if both instances of the group buffer pool are damaged. Automatic recovery is not needed for group buffer pools that are defined as GBPCACHE (NO).

Simplexed SCA and lock structure failures: To recover a simplexed SCA and lock structure, DB2 uses information that is contained in its virtual storage to quickly rebuild the structures. This has a minimal impact on performance.

DB2 can rebuild a simplexed SCA and lock structure in the same coupling facility or in an alternate coupling facility, assuming that the following conditions are true:

- You specified the alternate coupling facility in the CFRM policy preference list.
- You allocated enough storage in the alternate coupling facility to rebuild the structures there.

If DB2 fails to rebuild the SCA and lock structure from virtual storage, all active members in the group terminate abnormally, and you must perform a group restart to recover the necessary information from the logs.

Duplexed SCA and lock structure failures: If a duplexed SCA or lock structure fails, the failing structure is deallocated, and processing continues on the running structure.
Important: To enable DB2 to switch to the secondary SCA and lock structure, the CFRM policy for the structures must indicate that duplexing is allowed, and the SCA and lock structure must currently be running in duplexed mode.

Preparing for connectivity failures
DB2 and z/OS interpret a total failure of the coupling facility as a connectivity failure. This includes such issues as power failures to the coupling facility or a problem with coupling facility control code. See “Problem: all members have lost connectivity” on page 173 for more information about recovery from lost connections.

Rebuilding structures when connectivity is lost: As with structure failures, DB2 can recover quickly from connectivity failures if group buffer pool structures are duplexed, and the SCA and lock structure are simplex.

Recovering from connectivity failures with duplexed structures: DB2 recovers from connectivity failures with a duplexed group buffer pool, SCA, and lock structure by switching to the structure with good connectivity. Be aware that recovery of duplexed group buffer pool structures is quick and non-disruptive, but recovery of a duplexed SCA and lock structure can result in a significant impact on performance.

Recovery from connectivity failure with simplex structures: DB2 recovers from connectivity failures with a simplex group buffer pool, SCA, and lock structure as described below. Be aware that recovery of a simplex SCA and lock structure is fast and has little or no impact on performance, but recovery of simplex group buffer pool structures is very disruptive to the system.

When connectivity is lost, DB2 rebuilds simplex structures on the alternate coupling facility that is specified in the CFRM policy. In rebuilding these structures, DB2 attempts to allocate storage on the alternate coupling facility. DB2 uses the current size of the structure for the initial size of the structure on the alternate coupling facility. If DB2 cannot allocate the storage for the SCA or lock structure, the rebuild fails. If z/OS cannot allocate the storage for the group buffer pools, the changed pages are written to disk. See “Connectivity failure to the SCA or lock structure” on page 169 and “Connectivity failure to non-duplexed group buffer pools” on page 170 for more information about failure scenarios.

To control when a rebuild occurs, specify a rebuild threshold in the CFRM policy. This value is a percentage that you specify for the REBUILDPERCENT parameter. z/OS uses the REBUILDPERCENT value to determine whether to initiate a structure rebuild in the case of a loss of connectivity to the coupling facility that contains the structure. The percentage is based on the Sysplex failure management (SFM) weights of all the systems that have active connections to a structure at the time. (A connection is considered active even if connectivity is lost.) You also specify weights on the SFM policy.

z/OS calculates the total weight of:
- All systems that have active connections but that lost connectivity to the structure (A)
- All systems that have active connections to the structure (B)

z/OS then divides A by B and compares the result to the REBUILDPERCENT value. (In the case of multiple connections to a structure from a single z/OS system, that system is counted only once. Multiple connections from a single z/OS system exist, for example, when two members reside on the same z/OS system.)
The default value of REBUILDPERCENT is 100, which means that a structure is
not automatically rebuilt unless all the members that have active connections lose
connectivity to the structure. See “Problem: all members have lost connectivity” on
page 173 for more information about this situation.

Example: All z/OS systems in an eight-member Parallel Sysplex are of equal
weight (10), and six of the systems have active connections to the group buffer
pool. A equals the total weight of all systems that lose connectivity to the group
buffer pool. B equals the total weight of all systems that have active connections to
the group buffer pool. If two of the six systems lose connectivity, the value of A is
20, and the value of B is 60.

z/OS determines whether to initiate a rebuild as follows:
• If \((A/B) \times 100\) is greater than or equal to the REBUILDPERCENT value, DB2
rebuilds the structure.
• Otherwise, DB2 does not rebuild the structure. Instead, the affected members
disconnect from the group buffer pool.

In this example, \((20/60) \times 100\) is equal to 33. If the value of REBUILDPERCENT is
35, z/OS does not initiate a rebuild. See “Problem: a subset of members have lost
connectivity” on page 174 for more information about this situation.

Recommendation: If you have high availability requirements, specify a small
REBUILDPERCENT value, and allow the system to rebuild the structure. Specify a
larger REBUILDPERCENT value only if you prefer to have a member lose the use
of a structure rather than temporarily disrupt all of the members that were using
the structure so that the structure can be rebuilt. For more information, see z/OS
MVS Setting Up a Sysplex.

Monitoring rebuild events
You can monitor a performance class 20 trace (IFCID 0267 and 0268) to determine
how long a rebuild of a structure takes and the reason for the rebuild. You can also
examine the messages returned to the console to monitor a rebuild. The reasons for
a rebuild can include:
• Lost connectivity
• Operator command
• Duplexing is being established

Coupling facility volatility
There are times when a coupling facility enters a volatile state, which means that if
the power fails, data in the coupling facility at the time of the failure is not saved.
If the coupling facility is configured to be non-volatile (using the proper power
backup, such as a battery backup), volatility is generally a transient state that
might occur, for example, if you take the battery out.

DB2 issues a warning message if allocation occurs in a volatile coupling facility. A
change in volatility after allocation does not affect existing structures.

Advantages of a non-volatile coupling facility: If you lose power to a coupling
facility that is configured as non-volatile, the coupling facility enters power save
mode and saves the data contained in the structures. When power is returned, you
do not need to do a group restart nor recover the data from the structures. For
systems that require high availability, non-volatile coupling facilities are
recommended.
Preparing for coupling facility channel failure

To prevent a failure to DB2 that is caused by a coupling facility channel failure, consider using dual channels (sometimes called links) between each CPC and a coupling facility. Without dual links, a channel failure is more likely to occur than a failure in the coupling facility. Losing connectivity to the SCA or lock structure can bring that particular member down, unless you specify duplexing or an alternative coupling facility in the CFRM policy preference list.

Duplexing structures

Running some or all of the SCA, lock, and group buffer pool structures in duplex mode is one way to achieve high availability for these structures across many types of failures, including lost connections and damaged structures.

Recommendation: Although the SCA and lock structure can run in duplex mode, you should run them in simplex mode. When duplexed, the SCA and lock structure can cause a significant impact on performance. Also, because the SCA and lock structure can be quickly rebuilt, duplexing them does not add significant availability.

Important: If you decide to duplex the SCA and lock structure, you must duplex both structures to achieve the availability benefits. Duplexing only the SCA or only the lock structure does not provide any benefit.

The remainder of this section applies only to duplexing group buffer pool structures.

How group buffer pool duplexing works

With a duplexed group buffer pool structure, two allocations of the same structure use one connection from each member. Each structure allocation must be in a different coupling facility. z/OS prefers to place the structures in coupling facilities that are failure-isolated from one another.

Recommendation: Ensure that at least one of the group buffer pool structures is in a non-volatile coupling facility. If power is lost to both coupling facilities and both coupling facilities are volatile, you must recover the group buffer pool from the logs.

Characteristics of primary and secondary structures: z/OS commands let you stop and start duplexing, and let you choose which of the duplexed structures is the primary and which is the secondary.

- The primary structure is the one from which changed data is cast out to disk. DB2 also uses the primary structure to read data, keep track of page-registration, and perform cross-invalidation of pages in the buffer pools of individual members.
- When changed data is written to the primary structure, it is also written to the secondary structure. DB2 uses the secondary structure as a backup. After data is cast out to disk from the primary structure, the data is deleted from the secondary structure.

From a z/OS perspective, duplexing is really an extended rebuild, so z/OS documentation and commands sometimes call the primary structure the old structure, and refer to the secondary structure as the new structure.
Coupling facility storage considerations for duplexing: When planning for storage, make the primary and secondary group buffer pool structures the same size. If you are already properly configured for high availability using a simplex group buffer pool, you usually do not need extra coupling facility storage for duplexing. For simplex group structures, you must reserve enough spare capacity in the coupling facilities to be able to absorb the structures of any failed coupling facility. With duplexed structures, instead of reserving storage in case of a coupling facility failure, that storage is used by the secondary group buffer pool.

Example: Assume you have two coupling facilities, each with 1 GB of memory, for a total of 2 GB of memory. Ensure that the total size of the structures across the two coupling facilities does not exceed 1 GB (50% of the total coupling facility storage).

If you configure three or more coupling facilities, you might need an additional coupling facility for duplexing.

Requirements
For duplexing to work, the following conditions must be true:
- The CFRM policy preference list for the group buffer pool structure that you want to duplex must contain at least two coupling facilities with a CFLEVEL of 5 or higher. All members of the data sharing group must have physical connectivity to both coupling facilities in which the primary and secondary structures reside.
  - If you are going to do automatic reduplexing, you need three coupling facilities that are physically connected to members of the data sharing group.
- At least one member must be actively connected to the group buffer pool structure.
- The group buffer pool must be defined with GBPCACHE(YES), the default.

Establishing duplexing
You have three options on the CFRM policy for duplexing.
- DUPLEX(ENABLED) automatically starts duplexing.
- DUPLEX(ALLOWED) is not automatic, and you must issue a command to start duplexing.
- DUPLEX(DISABLED) disables duplexing.

For more information about starting and stopping duplexing, see "Starting duplexing for a structure" on page 196 and "Stopping duplexing for a structure" on page 196.

Performance of duplexing
The process of establishing duplexing can be somewhat disruptive because access to the group buffer pool structure is quiesced while the secondary structure is allocated. Also, changed pages are copied from the primary structure to the secondary structure (or cast out to disk). Transactions that need access to the group buffer pool structure during this process are suspended until the process is complete. Because of this disruption, you need to establish duplexing at a time of low activity on the system. How long the process takes depends on how many pages are copied to the secondary group buffer pool structure.

In general, more processor and elapsed time is needed to do duplexed group buffer pool structure writes and castout processing than to do simplex group buffer pool structure writes and castout processing. Workloads that are more update-intensive will probably experience a slight increase in host CPU usage.
when duplexing is activated. In most cases, the majority of the CPU increase occurs in the DB2 address space. Duplexing can cause a slight increase in the transaction elapsed time. Read performance is unaffected by duplexing.

You will also see an increase in the CPU usage in the coupling facility that contains the secondary structure. You can estimate approximately how much the coupling facility CPU usage will increase when you establish duplexing as follows:

1. Determine the amount of coupling facility CPU usage that the simplexed primary structure consumes.
2. Divide the result in half to determine how much coupling facility CPU usage the duplexed secondary structure will consume.

Duplexing should have little or no impact on the CPU usage in the coupling facility that contains the primary structure.

The statistics and accounting trace classes contain information about structure duplexing.

**Monitoring duplexing rebuilds**: When a group buffer pool structure is duplexed, it is considered to be in an extended rebuild status called a duplexing rebuild. This activity is reported in IFCIDs 0267 and 0268, along with other reasons for rebuilding.

**DB2 resource availability considerations**

DB2 resource availability considerations for a data sharing group are basically the same as for a single DB2 subsystem. This section describes high availability options for the catalog and directory, for data in group buffer pools, and for DB2 restart.

**Critical DB2 data**

Critical DB2 data should be placed behind high-availability storage controllers, such as the IBM Enterprise Storage Server®. Consider placing the catalog and directory behind a 3990 control unit with dual-write capability for hardware duplexing. Another possibility is to use storage controllers with high-availability characteristics (such as controllers that use RAID technology) for critical DB2 data.

**Group buffer pool data**

Assign data that require high availability to group buffer pools that reside on non-volatile coupling facilities. Use group buffer pool duplexing, which has minimal impact on performance and can avoid hours of recovery time.

**Data availability at restart**

For faster restarts, take more frequent checkpoints. Checkpoint frequency is the most important factor in DB2 restart time.

Make DB2 restart faster by enabling fast log apply. Enable fast log apply by specifying some amount of storage on the LOG APPLY STORAGE parameter of installation panel DSNTIPL.

If your installation sometimes has problems with units of recovery (URs) that take a long time to back out after a failure, make plans to reroute work to other members of the group. Another solution is to postpone backout processing for those long-running URs until DB2 is up and receiving new work. See the section "Postponing backout processing" on page 193 for more information.
Planning for WebSphere coordinated XA transactions

For XA connections coordinated by a remote Websphere Application Server and the DB2 Universal JDBC Driver, you are required to set up dynamic virtual IP addresses. A dynamic IP address needs to be configured for the DB2 group, and one address must be set up for each DB2 member in the group. Refer to Chapter 4, “Communicating with data sharing groups,” on page 107 for details on the configuration of dynamic VIPA. For XA recovery to be processed on any member after failure, information on indoubt XIDs must be obtained from the SCA. The SCA contains the list of indoubt XIDs for the entire group, as well as each member’s dynamic VIPA and the resync port of the member that owns the indoubt thread. XA recovery returns the list of all prepared or heuristically committed threads for the entire group back to the XA Transaction Manager (TM). The Java Universal Driver uses the information to send the commit decision to the owning member, when the XA TM recovers, and then commits or rolls back indoubt XIDs.

For the recovery process to function properly, you must set up the group dynamic VIPA. Figure 17 on page 46 illustrates how the XA recovery process, that is described above, works. The JDBC Universal Driver is looking to recover any indoubt XIDs. In this example, the JDBC driver must recover the XID indoubt that is in the log of the DB1G z/OS database. The SCA coupling facility contains the indoubt list for all three of the databases for this system. The SCA contains the indoubt XID in the DB1G log, and the indoubt XID is recovered by the JDBC Universal Driver.
When migrating from Version 7 to Version 8 new function mode using the DB2 Universal JDBC Driver, you must drop the external indoubt table. When you switch from compat mode to new function mode in Version 8, all of the indoubt XIDs automatically are stored in the SCA coupling facility, which replaces the external indoubt table used in Version 7. You can drop the external indoubt table manually or through the JDBC program, DB2T4XAIndoubtUtil. Verify all indoubt XIDs are resolved before dropping this table.

**Estimating storage**

This section gives you information about estimating storage for coupling facility structures and for DB2 resources. The following topics are described:

- “General information about coupling facility storage” on page 47
- “Group buffer pool sizes” on page 48
- “Lock structure size” on page 54
- “SCA size” on page 55
- “Changing structure sizes” on page 56
- “Estimating additional storage for IRLM” on page 57
- “Storage for DB2 objects” on page 58
General information about coupling facility storage

Precise estimates for the sizes of coupling facility structures are hard to provide. Every environment is different, and storage allocation is affected by the processor model and the level of coupling facility control code. Use the information in this section to estimate initial size (INITSIZE) values. Depending on how much your work load varies, consider specifying a larger value for SIZE.

For duplexed structures, the SIZE and INITSIZE parameter values apply to both instances of the structure.

The information in this section assumes that all page sets in a particular group buffer pool are defined with the same GBPCACHE attribute. You can put page sets with different GBPCACHE attributes in the same group buffer pool, but you must adjust the formulas accordingly.

Coupling facility structure size allocation

When a new coupling facility structure is allocated for a data sharing group, its size is usually taken from the value of the INITSIZE parameter in the CFRM policy.

After the structure is allocated, you can dynamically change its size with the z/OS command SETXCF START,ALTER,SIZE=newsize,STRNAME=strname. The new SIZE value cannot be greater than the SIZE value in the CFRM policy, but it can be smaller than the INITSIZE value. If the coupling facility has enough space, z/OS increases (or decreases) the size of the structure to the new size; the INITSIZE value of the policy remains unchanged.

In most cases, DB2 uses the new SIZE value for any subsequent allocations of the structure instead of the INITSIZE value in the CFRM policy. Any of the following subsequent allocations can use the new SIZE value:

- A group buffer pool or SCA is deallocated and then reallocated
- A secondary structure is allocated for a duplexed group buffer pool, SCA, or lock structure, if duplexing is started after the size of the primary structure was dynamically changed
- Any structure is rebuilt with the z/OS command SETXCF START,REBUILD,STRNAME=strname

The new size is recorded across a restart of DB2 and is used for all subsequent allocations until one of the following events occurs:

- A CFRM policy is started, and the policy has a different INITSIZE value than the size of the structure that was dynamically changed with the SETXCF START,ALTER command
- Another SETXCF START,ALTER command is issued to dynamically change the size of the structure

Exception: If a lock structure is deallocated and all the members are down, the INITSIZE value is used. This is a consideration for disaster recovery or for situations where data sharing groups are cloned. During normal operation of a data sharing group, you are unlikely to encounter this situation.

For more information on changing structure sizes, see “Changing structure sizes” on page 56.
**Coupling facility structure sizer:** The coupling facility structure sizing tool (CFSizer) simplifies the task of estimating coupling facility structure storage size by asking you questions about your existing or planned configuration. If you are not sure how to answer the questions, the help functions direct you to the source of the data on your system. Access the CFSizer at www.ibm.com/s390/cfsizer.

After you decide what your structure sizes are, include those values in the CFRM policy definition. See z/OS MVS Setting Up a Sysplex for more information about creating CFRM policies.

**Do not overestimate the SIZE parameter**

Coupling facility structures contain some static control structures. When a structure is initially allocated, these static structures are allocated to accommodate the potential size of the coupling facility structure. In other words, the size of the static structures is proportional to the maximum size (the value of the SIZE parameter) of the coupling facility structure. If the SIZE value is much larger than the INITSIZE value, a large percentage of the initial structure’s size might be used for these static structures, leaving you with little usable storage space in the structure.

**Recommendation:** In general, specify a SIZE value that is larger than the INITSIZE value, but limit the SIZE value in the following ways:

- Limit the SIZE value to no more than two to three times the INITSIZE value for the SCA and lock structure.
- Limit the SIZE value to no more than four times the INITSIZE value for group buffer pools.

For example, if the INITSIZE value for a group buffer pool is 100 MB, specify a SIZE value of 400 MB or less.

**Group buffer pool sizes**

A group buffer pool consists of two parts: data pages (sometimes called *data entries*) and directory entries.

**Data pages:** Data pages reside in the group buffer pool. The size of a data page is the same as the page size supported by the corresponding DB2 buffer pools (4 KB, 8 KB, 16 KB, or 32 KB).

If you are caching changed data only, you need enough space to cache changed data plus extra space for pages that are frequently referenced. By caching those frequently referenced pages in the group buffer pool, you can decrease the amount of time it takes for any member to refresh that page in its member buffer pool because you avoid the disk I/O.

If you choose GBPCACHE NONE or GBPCACHE SYSTEM, no user data pages are actually stored in the group buffer pool. However, with GBPCACHE SYSTEM, space map pages for LOBs are cached in the coupling facility.

**Directory entries:** A directory entry specifies the location and status of a page image somewhere in the data sharing group, whether the image is in the group buffer pool or in one of the member buffer pools. Only one directory entry exists for any given page, no matter how many places that page is cached.

The size of a directory entry is approximately 200 bytes, but it varies somewhat based on the size of the data pages and the CFLEVEL you are using. See *IBM eServer zSeries Processor Resource/System Manager Planning Guide* for the exact size.
**Specifying a ratio:** The amount of space that is allocated for a group buffer pool is determined by the ratio of directory entries to data pages. When you originally define a structure in the CFRM policy for a group buffer pool, you specify its total size. For DB2, the ratio defaults to five directory entries per data page. Later, you can change the ratio with the ALTER GROUPBUFFERPOOL command. The new value that you define takes effect when the group buffer pool is rebuilt or reallocated. See “Determining the correct size and ratio” on page 263 for information about detecting problems with the size and ratio of group buffer pools.

For group buffer pools defined with GBPCACHE(NO), ratios are ignored because no data is actually stored in the group buffer pool.

When possible, both a formula and a general guideline are provided to help you estimate the initial sizes and ratios of your group buffer pools. (The exception is for GBPCACHE ALL group buffer pools, for which only a general guideline is provided.)

The formula is not too complex and is likely to be more accurate, assuming that you are fairly confident of the values for the variables in the formulas. Otherwise, use the general guidelines and then adjust your values from there.

**Storage estimate for group buffer pools that cache changed data**

The size of a group buffer pool is related to the amount of sharing and the amount of updating. An estimate must be based on the total amount of member buffer pool storage multiplied by a percentage based on the amount of update activity. As data sharing and updating increases, more pages must be cached in the group buffer pool, and more directory entries are needed to track inter-DB2 interest.

**Formula:** The formula for estimating storage for group buffer pools that cache changed data is:

\[
\begin{align*}
\text{Data} &= U \times D \times R \\
\text{Data(MB)} &= \text{Data} / 1024 \\
\text{Dir} &= \text{Data} + (U \times VP) \\
\text{Dir(MB)} &= \text{Dir} / 1024 \\
\text{GBP(MB)} &= \text{Data(MB)} + \text{Dir(MB)} \\
\text{RATIO} &= \text{Dir(MB)} / \text{Data} \\
\end{align*}
\]

Where:

- **U** The estimated degree of data sharing:
  - 1 A high amount of sharing with a lot of update activity
  - 0.7 A moderate amount of sharing with a moderate amount of update activity
  - 0.5 A low amount of sharing with a low amount of update activity

- **D** The number of data pages written to disk per second for all members during peak activity. Do not use the number of pages written to the group buffer pool; D must be a count of distinct pages. To determine this value, use the field QBSTPWS from IFCID 0002 (the PAGES WRITTEN field of the buffer pool section of the OMEGAMON Statistics report.)

- **R** The average page residency time in the group buffer pool, in seconds. This value is application-dependent, but you can assume that the typical range is 30 to 180 seconds. If you have no information about residency time, use 120.
**General guideline:** Make \( R \) large enough so that other members can refresh an invalidated local copy of a changed page. When a changed page is written to the group buffer pool, it invalidates local copies of the page in other members. The changed page needs to remain resident in the group buffer pool long enough for other members to refresh the page.

- \( P \) The page size (4 KB, 8 KB, 16 KB, or 32 KB).
- \( VP \) The number of data pages defined for the virtual pool (the sum across all of the members).

- \( 0.2 \) The approximate size of a directory entry, in KB.
- \( 1.1 \) The additional storage needed for coupling facility control structures.

**Example:** Assume that you have a two-member data sharing group for which you have determined the following information:
- The degree of data sharing is very high (\( U = 1 \)).
- 500 disk writes per second occur across both members (\( D = 500 \)).
- No information exists about the page residency time (\( R = 120 \)).
- The page size is 4 KB (\( P = 4 \)).
- Member 1 is configured with a virtual pool of 80000 buffers.
- Member 2 is configured with a virtual pool of 40000 buffers (\( VP = 120000 \)).

The calculation is as follows:

```plaintext
# Data_entries = 1 * 500 * 120 = 60000
# Data(MB) = 60000 * 4 / 1024 = 234 MB
# Dir_entries = 60000 + (1 * 120000) = 180000
# Dir(MB) = 1.1 * 180000 * 0.2 / 1024 = 38 MB
# GBP(MB) = 234 MB + 38 MB = 272 MB
# RATIO = 180000 / 60000 = 3.0
```

The preceding calculation indicates that the group buffer pool should be defined with an INITSIZE of 272 MB. Use the ALTER GROUPBUFFERPOOL command to change RATIO to 3.

**General guideline:** For installation planning purposes, you should use the following general guideline as an initial estimate for the size of a DB2 group buffer pool for table spaces, indexes, or partitions that cache only changed data (GBP(CACHE(NEW))):

Add the local buffer pool storage for this buffer pool number across all the members of the group. Then, multiply this amount by one of these workload factors:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>For light sharing with a low amount of updating activity</td>
</tr>
<tr>
<td>20%</td>
<td>For medium sharing with a moderate amount of update activity</td>
</tr>
<tr>
<td>40%</td>
<td>For a high amount of sharing with a lot of update activity</td>
</tr>
</tbody>
</table>

You can run a trace for IFID 0002 to obtain an estimate of the amount of data sharing in your system. Calculate the “degree of data sharing” by dividing QBGLGG, the number of get pages for group buffer pool-dependent objects, by QBSTGET, the number of get pages. A value that is less than 25% is considered to be light data sharing, a value between 25% and 75% is medium data sharing, and a value greater than 75% is high data sharing.
Remember that the type of workload you run can influence the amount of storage you use. For example, if you have “hot spots” in which updates to a single page are frequent rather than spread throughout the table space, you might need less storage for caching.

Example: Assume that the total buffer pool storage for all the members of the group is 400 MB, and you expect a medium amount of read/write sharing in the environment. The following calculation is an estimate for the size of a group buffer pool when GBPCACHE(CHANGED):

\[400 \text{ MB} \times 20\% = 80 \text{ MB}\]

**General guideline for storage estimate for caching all data**

For installation planning purposes, use the following as an initial estimate of the size of a DB2 group buffer pool when the installation caches read-only pages with changed pages (GBPCACHE(ALL)).

Calculate the sum of the local buffer pool storage for this buffer pool number (virtual only) across all members in the group. Then, multiply this amount by one of these workload factors:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>50%</td>
<td>Less than half of the table spaces, indexes, or partitions specify GBPCACHE ALL</td>
</tr>
<tr>
<td>75%</td>
<td>Half of the table spaces, indexes, or partitions specify GBPCACHE ALL</td>
</tr>
<tr>
<td>100%</td>
<td>75% of more of the table spaces, indexes, or partitions specify GBPCACHE ALL for a high sharing environment</td>
</tr>
</tbody>
</table>

Example: Assume that the local buffer pool storage on all members of the group adds up to 200 MB. Half of the page sets that come into the pool are defined as GBPCACHE(ALL). The following calculation is an estimate for the size of a group buffer pool when GBPCACHE(ALL):

\[200 \text{ MB} \times 75\% = 150 \text{ MB}\]

**Storage estimate for caching no data**

The following is a formula for estimating storage for group buffer pools that cache no data (GBPCACHE(NO)):

\[
\begin{align*}
\text{Dir}_{-\text{entries}} &= U \times VP \\
\text{Dir} (\text{MB}) &= 1.1 \times \text{Dir}_{-\text{entries}} \times 0.2 / 1024 \\
\text{GBP} (\text{MB}) &= \text{Dir} (\text{MB}) \\
\text{RATIO} &= n/a
\end{align*}
\]

If the group buffer pool itself is defined with GBPCACHE(NO), the ratio is ignored.

The variables are the same as described in [“Formula” on page 49](#). In summary, they are:

- **U**: The estimated degree of data sharing.
- **P**: The page size (4 KB, 8 KB, 16 KB, or 32 KB).
- **VP**: The number of data pages defined for the virtual pool (the sum across all the members).
**Example:** Assume that you have a two-member data sharing group for which you have determined the following information:

- The degree of data sharing is very high (U=1).
- Member 1 is configured with a virtual pool of 80000 buffers.
- Member 2 is configured with a virtual pool of 40000 buffers (VP=120000).

The calculation is as follows:

\[
\begin{align*}
\text{Dir\_entries} & = 1 \times 120000 = 120000 \\
\text{Dir\_MB} & = 1.1 \times 120000 \times 0.2/1024 = 25 \text{ MB} \\
\text{GBP\_MB} & = 25 \text{ MB}
\end{align*}
\]

The preceding calculation indicates that the group buffer pool should be defined with an INITSIZE of 25 MB. Use the command `ALTER GROUPBUFFERPOOL` to change the GBPCACHE attribute to NO. If you put GBPCACHE NONE page sets in a GBPCACHE(YES) group buffer pool, the calculation becomes more complicated. You do this because the RATIO is observed and you are probably going to waste a lot of space on unneeded data entries.

**Storage estimate for caching LOB space maps (GBPCACHE SYSTEM)**

The following is a formula for estimating storage for group buffer pools that cache LOB space map data:

\[
\begin{align*}
\text{Data\_entries} & = (U \times D / 10) \times R \\
\text{Data\_MB} & = \text{Data\_entries} \times P / 1024 \\
\text{Dir\_entries} & = \text{Data\_entries} + (U \times VP) \\
\text{Dir\_MB} & = 1.1 \times \text{Dir\_entries} \times 0.2 / 1024 \\
\text{GBP\_MB} & = \text{Data\_MB} \div \text{Dir\_MB} \\
\text{RATIO} & = \text{MIN}(\text{Dir\_entries} / \text{Data\_entries}, 255)
\end{align*}
\]

The variables are the same as described in "Formula" on page 49. In summary, they are:

- **U** The estimated degree of data sharing.
- **D** The number of data pages that are written to disk per second for all members during peak activity. Do not use the number of pages that are written to the group buffer pool; D must be a count of distinct pages. To determine this value, use the field QBSTPWS from IFCID 0002 (the PAGES WRITTEN field of the buffer pool section of the OMEGAMONStatistics report).
- **10** An estimate of the LOB system pages that are written for every LOB data page.
- **P** The page size (4 KB, 8 KB, 16 KB, or 32 KB).
- **R** The average page residency time in the group buffer pool in seconds.
- **VP** The number of data pages that are defined for the virtual pool (the sum across all the members).

**Example:** Assume that you have a two-member data sharing group for which you have determined the following information:

- The degree of data sharing is moderate (U=.7).
- 10 disk writes per second occur across both members during peak activity (D=10).
- The space map page is resident in the group buffer pool page for 120 seconds (R=120).
• The page size is 32 KB (P=32).
• Member 1 is configured with a virtual pool of 20000 buffers.
• Member 2 is configured with a virtual pool of 10000 buffers (VP=30000).

The calculation is as follows:

\[
\begin{align*}
\text{Data\ entries} &= (\{.7 \times 10\} / 10) \times 120 = 84 \\
\text{Data(MB)} &= 84 \times 32 / 1024 = 2.6 \text{ MB} \\
\text{Dir\ entries} &= 84 + (\{.7 \times 30000\}) = 21084 \\
\text{Dir(MB)} &= 1.1 \times 21084 \times 0.2 / 1024 = 4.5 \text{ MB} \\
\text{GBP(MB)} &= 2.6 \text{ MB} + 4.5 \text{ MB} = 7.1 \text{ MB} \\
\text{RATIO} &= \text{MIN} (21084 / 84, 255) = 251
\end{align*}
\]

The preceding calculation indicates that the group buffer pool should be defined with an INITSIZE of 7.1 MB. The ratio is less than the maximum value, so use the command ALTER GROUPBUFFERPOOL to change the ratio to 251.

**PR/SM™ formulas for calculating sizes of group buffer pools**

You can also calculate group buffer pool sizes using the coupling facility allocation formulas for cache structures found in *IBM eServer zSeries Processor Resource/System Manager Planning Guide*. Table 5 contains information that is used in those formulas. The size of cache structures in DB2 can vary greatly based on the amount of data for which inter-DB2 read/write interest exists at any given time. You probably need to monitor the use of the group buffer pools and adjust their sizes accordingly.

**Table 5. Information for calculating cache structure sizes**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DB2 Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC</td>
<td>1</td>
<td>Maximum storage class</td>
</tr>
<tr>
<td>MCC</td>
<td>1024</td>
<td>Maximum castout class</td>
</tr>
<tr>
<td>MDAS</td>
<td>Dependent on page size being cached: 1 for 4 KB pages 2 for 8 KB pages 4 for 16 KB pages 8 for 32 KB pages</td>
<td>Maximum data area size</td>
</tr>
<tr>
<td>DAEX</td>
<td>4</td>
<td>Data area element characteristic</td>
</tr>
<tr>
<td>AAI</td>
<td>0</td>
<td>Adjunct assignment indicator</td>
</tr>
</tbody>
</table>

**R_de**

Set on ALTER GROUPBUFFERPOOL. Default is 5. See Table 6 on page 54 for more information on determining this value.

**R_data**

Set on ALTER GROUPBUFFERPOOL command. See Table 6 on page 54 for more information on determining this value.

Table 6 on page 54 contains information about the formulas used to determine R_data and R_de. N is the RATIO entered on the ALTER GROUPBUFFERPOOL command. If N has a decimal point, all digits after the first decimal place are ignored. For example, 5.67 is treated as 5.6. For more information about the ALTER BUFFERPOOL command, see Part 3 of [DB2 Command Reference](#).
Lock structure size

The coupling facility lock structure contains two parts. The first part is a lock entry table, which is used to determine if inter-DB2 read/write interest exists on a particular hash class. Hash classes are resources that hash to a particular place in the lock table. The second part is a list of the update locks that are currently held (sometimes called a modify lock list or record list table). You can control the division of the lock structure storage between these two components by using the IRLMPROC or by using an IRLM MODIFY command. If you do not specify how the structure is to be split, IRLM attempts to divide it with a 1:1 ratio between lock table entry (LTE) and record list entry (RLE) storage.

The total size of the lock structure must be large enough to limit hash contention, which prevents performance problems. The lock structure must also be large enough to prevent failures that result from a lack of record table storage to write a MODIFY entry (RLE). Proper specification for the number of LTEs can help avoid hash contention, as described in “Avoiding false contention” on page 223.

IRLM reserves 10% of the record table entries for “must complete” functions (such as rollback or commit processing), so that a shortage of storage does not cause a member to fail. However, if storage runs short in the record table, there can be an impact on availability (transactions are terminated), response time, and throughput. See “Monitoring DB2 locking” on page 229 and “Changing the size of the lock structure” on page 234 for more information.

Specifying the lock entry size

The LOCK ENTRY SIZE parameter on installation panel DSNTIPJ determines the amount of space required for lock contention control information (that is, individual entries in the lock table). The lock entry size and the number of lock table entries of the first IRLM to join the group determines the whole group’s storage size for the lock table and the lock table entry width. The default is 2 bytes, which is probably the size you want, unless you immediately create a data sharing group of seven or more members. By restricting each lock entry to 2 bytes, you maximize the amount of RLE space available from the define structure size.

Storage estimate for the lock structure

For installation planning purposes, the initial size of the lock structure is based on how much updating you do. Table 7 on page 55 gives you initial size value recommendations.

Recommendation: If you do not specify a value for the LTE= parameter in the IRLMPROC, choose a value for INITSIZE that is a power of 2. This enables IRLM
to allocate the coupling facility storage so that half is used for lock table entries and the remainder is used for record table entries. If a 1:1 split occurs and total size is not a power of 2, you might experience severe shortage of space for the record table entries, resulting in DB2 or possible IRLM failures. A failure will occur because the number of lock table entries requested on CONNECT must be a power of 2. The record table is susceptible to storage shortages if the structure is too small or if the allocation of the lock table leaves too little storage for the record table.

When specifying a value for the LTE= parameter in the IRLMPROC, or when issuing the MODIFY irlmproc SET,LTE= command, you should monitor XES contention rates to determine the optimum value for your normal operating environment. If the contention rates appear to be too high, increase the LTE= parameter value to the next power of 2. Remember, any increase in the size of the lock table will cause a corresponding decrease in the record table, unless the structure size is also increased.

If you have little contention and want more storage available for record table entries, decrease the LTE= parameter value by a power of two. Anytime the number of lock table entries is decreased, you should monitor contention rates for a period of time.

Because structure allocation is done at CONNECT, any change that is made to the LTE= parameter value does not take effect unless the group is terminated, structure forced and the group restarted, or a REBUILD is done. Also, the LTE= parameter value of the first IRLM to CONNECT dictates the coupling facility structure sizes that are used by the group.

<table>
<thead>
<tr>
<th>INITSIZE</th>
<th>SIZE</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 MB</td>
<td>32 MB</td>
<td>For light sharing with a low amount of updating, or for a single-member data sharing group</td>
</tr>
<tr>
<td>32 MB</td>
<td>64 MB</td>
<td>For medium sharing with a moderate amount of update activity</td>
</tr>
<tr>
<td>64 MB</td>
<td>128 MB</td>
<td>For a high amount of sharing with a lot of update activity</td>
</tr>
</tbody>
</table>

### SCA size

The shared communications area (SCA) is a list structure in the coupling facility that contains:

- Member names
- BSDS names
- Database exception status conditions about object in the database
- Recovery information, such as log data set names and the list of indoubt XA transactions

You can use the coupling facility structure sizing tool (CFSizer) to help you calculate structure sizes. This tool is located at: [www.ibm.com/s390/cfsizer](http://www.ibm.com/s390/cfsizer). Table 8 shows how to estimate the size of the SCA. The SCA size can be specified in 1 KB increments.

<table>
<thead>
<tr>
<th>Site size</th>
<th>Databases</th>
<th>Tables</th>
<th>INITSIZE</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>50</td>
<td>1000</td>
<td>16 MB</td>
<td>32 MB</td>
</tr>
</tbody>
</table>
Table 8. Estimating storage for the SCA (continued)

<table>
<thead>
<tr>
<th>Site size</th>
<th>Databases</th>
<th>Tables</th>
<th>INITSIZE</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>200</td>
<td>4000</td>
<td>32 MB</td>
<td>64 MB</td>
</tr>
<tr>
<td>Large</td>
<td>400</td>
<td>10 000</td>
<td>64 MB</td>
<td>128 MB</td>
</tr>
<tr>
<td>Extra Large</td>
<td>600</td>
<td>50 000</td>
<td>128 MB</td>
<td>256 MB</td>
</tr>
</tbody>
</table>

Running out of space in the SCA can cause DB2 to fail. Because much of the space in the SCA is taken up with exception information, you can reclaim space by correcting database exception conditions. If the data sharing group supports XA transactions, be sure to reserve additional space in the SCA for the indoubt transaction ID (XID) list. (DRDA XA protocol support enables DB2 servers to participate in WebSphere® coordinated two-phase commit transactions that have multiple connections to more than one data source.) After a failure, indoubt XA transactions are resolved by WebSphere, or you can reclaim space by resolving indoubt XA transactions manually. See DB2 Administration Guide for detailed instructions about using the DB2 command -RECOVER INDOUBT to manually resolve indoubt XA transactions.

**Changing structure sizes**

You can change the size of individual structures by changing the CFRM policy and then rebuilding the structures by using the z/OS command SETXCF START,REBUILD,STRNAME=servername. (Be aware that this command does not work on structures that are actively being duplexed.)

z/OS attempts to reallocate a new instance of the structure in the same coupling facility if that coupling facility has enough storage space. If there is not enough space, z/OS looks at the preference list and uses the specified alternate coupling facility. After the space is allocated, DB2 rebuilds the information into the new structure. Any transactions that need the structure must wait until the rebuild is complete. It is best to plan a rebuild when other activity in the system is low.

**Dynamically changing the structure size**

If the affected structure is allocated in a coupling facility with CFLEVEL greater than 0, you can dynamically increase the structure sizes up to a maximum limit specified on the CFRM policy by using the z/OS command SETXCF START,ALTER.

The advantages to this method are:

- DB2 can access the structures while a change is taking place.
- Less coupling facility storage is required because DB2 does not need to allocate enough space for a whole new structure. It dynamically adds or deletes storage from the existing structure.
- The command works on duplexed structures.

For more information on changing structure sizes, see:

- “Changing the lock structure size dynamically” on page 234 for the lock structure
- “Problem: storage shortage in the SCA” on page 179 for the SCA
- “Dynamic method” on page 270 for the group buffer pool
- “Estimating storage” on page 46 for implications concerning size allocations
Allowing structures to be altered automatically

z/OS can automatically alter a coupling facility structure when the structure reaches an installation-defined or default percent-full threshold, as determined by structure full monitoring. The alter process can change the size of the structure, reapportion the objects within the structure, or both. This function works the same in data sharing environments as it does in non-data sharing environments. See z/OS MVS Setting Up a Sysplex for complete information about allowing structures to be altered automatically.

Estimating additional storage for IRLM

The requirements for IRLM lock storage are described in Part 2 of DB2 Installation Guide.

For data sharing, plan for additional storage to accommodate data sharing specific locks called P-locks. These locks are held on open page sets and on database descriptors (DBDs), skeleton cursor tables (SKCTs), and skeleton package tables (SKPTs). Unlike transaction locks, storage for P-locks is held even when no transaction activity exists; therefore they consume storage even with no transaction activity. See "Using P-locks" on page 243 for more information about P-locks.

Plan also for the extra storage that IRLM needs to build retained locks in case other members fail. Table 9 shows the variables you need to take into account.

Table 9. Variables used to estimate additional IRLM storage

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>P-locks</td>
<td>N = (MAX_OPEN_DATA_SETS × 500)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X = (N + (N × .4))</td>
</tr>
<tr>
<td>Y</td>
<td>Ability to hold update retained locks for a failed member</td>
<td>Depends on the update-intensity of the workload.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start with the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y = .25(X)(X)</td>
</tr>
</tbody>
</table>

Note:
1. The formula assumes that more than one P-lock might be held on a page set occasionally (such as for castout activity), and estimates about 40% for P-locks on the EDM pool objects and for short-lived page P-locks. If you know that the EDM pool has relatively few objects in it, you can use a lower percentage for that value. See Part 5 (Volume 2) of DB2 Administration Guide for more information about estimating the maximum number of open data sets, or use the value specified for the subsystem parameter DSMAX.

Setting a high priority for IRLM

Follow the guidelines documented in Part 5 (Volume 2) of DB2 Administration Guide for setting the priority of IRLM when using workload manager. If IRLM priority is too low, storage might not be freed as quickly, and IRLM might run out of storage.

The IRLM address space priority, though it should remain high, should be lower than XCFAS.

Monitoring IRLM storage use

Use the z/OS command MODIFY irlmproc,STATUS,STOR to see how much storage IRLM is using. For more information about the syntax of this command, see Part 3 of DB2 Command Reference.
Increasing IRLM storage for Sysplex query parallelism

Sysplex query parallelism uses IRLM Notify messages to pass data between the assistants and the coordinator:

- The coordinator uses Notify messages to communicate with the assistants.
- The assistants use Notify messages to communicate with the parallelism coordinator.

Before you use Sysplex query parallelism, make sure that you have enough ECSA to handle these messages.

Calculating storage for the coordinator: Any member that can be a coordinator needs approximately 200 KB of extra storage for messages that it sends to the assistants.

Calculating storage for the assistants: An assisting DB2 is any member that receives query work from another member of the data sharing group. To estimate the amount of extra storage that IRLM requires on an assisting member:

1. Estimate the number of parallel tasks that can run concurrently on the assisting member.
2. Divide that number by the number of members sharing the query work.
3. Multiply the result by 32 KB.

\[(\text{numbers of queries} \times \text{max concurrent tasks}) \div \text{number of members}) \times 32 \text{ KB}\]

Example: Assume that you have a data sharing group in which all four members participate in processing parallel queries. If you have a total of 10 queries executing concurrently, and the highest number of parallel tasks is approximately 40, the calculation is:

\[(10 \times 40) \div 4 = 100\]

\[100 \times 32 \text{ KB} = 3 \text{ MB of extra storage on assistant}\]

To estimate the number of parallel tasks, you can look at EXPLAIN output or instrumentation records for the concurrent queries.

Storage for DB2 objects

General guideline: You can use the storage estimates in Part 2 of \[DB2\] Installation Guide for your capacity planning. This section describes some additional information that is specific to data sharing.

Estimating storage for the EDM pool

The formula you use to calculate storage for the environmental descriptor manager (EDM) pool is described in Part 2 of \[DB2\] Installation Guide. For data sharing, you might need to increase that storage estimate by about 10% because of the way DB2 cross-invalidates database descriptors (DBDs). This percentage is just an estimate; the actual amount of the increase depends on how often you create, drop, and alter objects in the data sharing group.

Cross-invalidating items in the EDM pool: DB2 does not have a backup EDM pool in the coupling facility for invalidating objects in the EDM pool (DBDs, cursor tables, and more) because these objects are modified less frequently than database data. So, there is one EDM pool for each member. When a DBD changes, DB2 uses XCF messages to notify other members which are also using that DBD that new transactions should use the new DBD, which is read into the EDM pool. Thus, it is
possible that one transaction is using the new DBD while other, currently running transactions are still using the old DBD. In other words, more than one copy of a DBD can exist in the EDM pool.

Reducing the storage impact: For CREATE, ALTER, or DROP statements, the DBD is not modified until a COMMIT is issued. You can significantly reduce the number of EDM versions by issuing CREATE, ALTER, or DROP statements within a single COMMIT scope. However, the exclusive lock on the DBD is held until the COMMIT.

# Before you enable DB2 data sharing

You can move to DB2 data sharing in one of three ways:
- Install Version 8 as new, and then enable data sharing.
- Migrate from a Version 7 non-data sharing environment to Version 8, and then enable data sharing.
- Migrate from a Version 7 data sharing environment to Version 8.

This process is described more fully in “Enabling DB2 data sharing” on page 80. If you already have a data sharing group, see “Migrating an existing data sharing group to a new release” on page 91 for information about migrating that group.

After you enable data sharing, disabling data sharing is a very difficult process and one which you should avoid. See “Disabling and re-enabling DB2 data sharing” on page 102 for more information.

This section describes considerations for planning your move to data sharing:
- “Deciding if merging is the right thing to do” on page 60
- “Connecting IMS and CICS” on page 61
- “Registering command prefixes and member group attachment name” on page 61
- “Specifying the group attachment name” on page 61
- “Increasing the size of the BSDS” on page 63
- “Increasing the size of the SYSLGRNX table space” on page 63

See also “Tuning deadlock and timeout processing” on page 226 for information about tuning your timeout periods.

Deciding if merging is the right thing to do

Although DB2 cannot automatically merge catalogs, you can merge existing DB2 subsystems into a data sharing group. Consider carefully a decision to merge existing DB2 subsystems.

Merging is a very complicated process. It involves not only the physical issue of moving data, but also many other management issues, including:
- Naming conventions for users, plans, packages, databases, tables, views, and more
- Authorization techniques
- Backup and recovery conventions
- Availability practices
Before you consider merging existing DB2 subsystems into a single data sharing group, ask yourself the following question: Why are the DB2 subsystems separate now?

**Reasons not to merge**
Reasons not to merge DB2 subsystems into data sharing groups include:

- If the DB2 subsystems are currently separate because different sets of user groups do not need frequent access to each other’s data, do not merge the subsystems.
- For the same reasons that you do not include test and production DB2 subsystems in a single z/OS system, **do not** merge test and production subsystems into a single data sharing group.
- If the security needs of the DB2 subsystems are different, do not merge the subsystems.
- If you want to minimize the number of DB2 subsystems because you do not have enough subsystem recognition characters, use 8-character command prefixes for relief instead of merging the subsystems.
- If you have two existing DB2 subsystems, and each of those subsystems can grow into a separate group, availability is usually better if you keep those groups separate. Administration is simpler if you keep groups split along the same lines as users.

**Reasons to merge**
If the subsystems are currently separate because of capacity constraints only, merging might be a good idea. Merging is good especially if the subsystems already share a common naming convention.

If the subsystems have, or need, a lot of common data and use shared read-only data, distributed access, or data replication to accommodate the sharing of data, merging might be a solution. However, this might not be a good approach if the security needs of the two subsystems are different. If you try to merge two subsystems with different security needs, especially if a shared naming convention is not already in place for those separate subsystems, merging them could be difficult.

**Connecting IMS and CICS**
You must define IMS and CICS connections for each member of a data sharing group. See Part 2 of [DB2 Installation Guide] for more information about connecting IMS and CICS to DB2.

**Connecting CICS to DB2**
The CICS attachment facility command, DSNC STRT, lets you override the subsystem name on startup. See Part 3 of [DB2 Command Reference] for more information about CICS attachment facility commands.

**Connecting IMS to DB2**
For every member that runs IMS applications, make sure that you attach IMS to that member. IMS must include a separate member SSM for every member DB2. See Part 2 of [DB2 Installation Guide] for more information about connecting IMS to DB2.
Registering command prefixes and member group attachment name

Use parameter library member IEFSSNxx to register the 1- to 8-character command prefix for a member and the group attachment name for the group. For example:

```
SUBSYS SUBNAME(ssname)
INITPARM('DSN2INI)
INITPARM('DSN3EPX, prefix<,scope<,group-attach>>'
```

**Recommendation:** When you register the command prefix in parameter library member IEFSSNxx and specify the scope of the prefix, choose a scope of started (S). This specification lets all z/OS systems in a single PARMLIB member IEFSSNxx use all z/OS systems in the Parallel Sysplex. It also simplifies the task of moving a member from one system to another; you can stop DB2 on one z/OS system and start it on another. You do not need to re-IPL the system.

For more detail about parmlib member IEFSSNxx, see information about job DSNTIJMV in Part 2 of [DB2 Installation Guide](#).

**Sample definitions**

The following sample definitions might appear in the shared parameter library SYS1.PARMLIB:

```
DB1A,DSN3INI,'DSN3EPX,-DB1A,S,080A'
DB2A,DSN3INI,'DSN3EPX,-DB2A,S,080A'
DB3A,DSN3INI,'DSN3EPX,-DB3A,S,080A'
DB4A,DSN3INI,'DSN3EPX,-DB4A,S,080A'
```

With these definitions, you can start DB1A on z/OS1, and that member is the only one in the Parallel Sysplex that can use -DB1A as its command prefix. However, because the member is registered with z/OS with a scope of S, you can stop DB1A and restart it on another z/OS without having to re-IPL any z/OS system.

**Changing the command prefix**

To change the command prefix parameters, you must change entry IEFSSNxx and re-IPL the host system. For example, if you want to change the command prefix scope from system-wide to Parallel Sysplex-wide, and you want to register the prefix at DB2 startup, change the M in the entry to S before you re-IPL.

If you want to use multiple-character command prefixes, make sure that your automation programs can handle multiple-character prefixes in messages before you change the prefixes.

**Specifying the group attachment name**

**Recommendation:** As shown in “Registering command prefixes and member group attachment name,” you should specify the group attachment name in member IEFSSNxx. You can let the DB2 installation process do this for you, or you can update the member yourself. Specify the group attachment name at a convenient time (during a planned IPL, for example).

Even if you have not yet enabled data sharing, the group attachment name is active after you IPL the system. An active group attachment name is not a problem. Until you are ready to move to data sharing, continue to specify the DB2 subsystem name in your TSO and batch jobs. When you are ready to move to data sharing, you can change those jobs to specify a group attachment name without the need for an IPL.
The group attachment name should not be the same as the subsystem names.

**How DB2 chooses a subsystem name:** When you submit a job on a z/OS system, DB2 treats the name that you specified on the DB2 connection request as either a subsystem name or a group attachment name. DB2 first assumes that the name is a subsystem name and attaches to that subsystem if either of the following are true:

- The subsystem is started.
- The subsystem is not started, and NOGROUP was specified in the DB2 connection request. NOGROUP indicates that group attach processing is not to be considered. If RETRY was specified in the command, DB2 tries to attach the subsystem again in 30 seconds. The value of RETRY determines the number of times that DB2 re-attempts to attach.

DB2 assumes that the name on the DB2 connection request is a group attachment name if no qualifying subsystem is found and either of the following are true:

- No subsystem with the name in the command is defined.
- A subsystem with that name is not started, the group attachment name is the same as its subsystem name, and NOGROUP was not specified in the DB2 connection request.

When DB2 assumes that the name is a group attachment name, it performs the following actions:

- Constructs a list of DB2 subsystems that are defined to this z/OS.
  
  To create the list, DB2 adds each subsystem when it goes through subsystem initialization. At IPL time, subsystems are initialized in the order in which they appear in member IEFSSNx.x. If you add a subsystem with the z/OS SETSSI command, that subsystem is added to the list at that time.

- Tries to attach to each subsystem in the order of the list until it finds one that is started on this z/OS or it reaches the end of the list.
  
  DB2 always attaches to the first subsystem on the list that is started—there is no load balancing.

If the name on the DB2 connection request is not a group attachment name, a “not started” message is returned.

**When a subsystem and a group attachment name are the same:** When you begin moving to data sharing, ensure that your definitions IEFSSNx.x are correct and DB2 connection requests are coded to get the results that you intend. You should ensure the definitions are correct, especially when the group attachment name is the same as a subsystem name. Incorrect definitions IEFSSNx.x might be troublesome if you have inactive subsystems that are still defined but not used.

**Example:** Assume you have the following subsystem definitions on a z/OS system:

```
DB2A,DSN3IN,,'DSN3EPX,-DB2A,S' *Inactive subsystem
DB1A,DSN3IN,,'DSN3EPX,-DB1A,S,0DB2A' *Active subsystem
```

The jobs submitted on this z/OS system try to connect to the name DB2A. DB2 tries to connect to subsystem DB2A before considering group attachment processing. However, because DB2A is not started and it lacks a group attachment name, DB2 does not invoke group attachment processing and find DB1A as you might have intended. To avoid this situation, include the group attachment name in the definition of DB2A, or remove entry IEFSSNx.x for subsystem DB2A if it is obsolete. With DB2A defined as the group attachment name for subsystem DB2A, DB2 tries to attach to DB1A after it discovers that DB2A is not started.
Alternatively, you might want a job to connect to a specific subsystem, but it connects to another subsystem in the group instead.

**Example:** Assume that you have the following subsystem definitions on a z/OS system:

DB1A,DSN3INI,'DSN3EPX,-DB1A,S, DB1A'.  +Inactive subsystem  
DB2A,DSN3INI,'DSN3EPX,-DB2A,S, DB1A'.  +Inactive subsystem  
DB3A,DSN3INI,'DSN3EPX,-DB3A,S, DB1A'.  +Active subsystem

Notice that DB1A is specified as a subsystem and also as the group attachment name for all three subsystems.

The jobs are submitted on this system with a DB2 connection request that specifies the name DB1A and omits the NOGROUP keyword. DB2 tries to connect to subsystem DB1A, but does not. Instead, DB2 invokes group attachment processing and eventually connects to DB3A, the first system with the group attachment name that is started. You might have intended that the job connect only to DB1A. To ensure that DB2 connects to subsystem DB1A and no other subsystem, specify NOGROUP in the DB2 connection request to disable group attachment processing. Specify the RETRY keyword in the request to indicate the number of times, at 30-second intervals, that DB2 will retry to connect to DB1A.

**Increasing the size of the BDSs**

Data sharing causes additional records to be written to the BDSs for member information.

**Recommendations:** To prevent the BDSs from expanding into secondary extents, change the size of the primary space allocation to 850 records. This is necessary only for members that have followed the IBM-recommended migration path from Version 7 without altering that record size. New installations and member installations already do this for you.

To increase the space allocation for the BDSs:

1. Rename existing BDSs.
2. Define larger BDSs with the original names.
3. Copy the renamed BDSs into the new BDSs.

You can do this using access method services. To see the definition used for the BDSs, see the installation job DSNTIJIN.

**Increasing the size of the SYSLGRNX table space**

The SYSLGRNX directory table space contains the relative byte address (RBA) ranges showing when data sets are open for updating. Because more members open and close data sets in a data sharing configuration, this table space is likely to grow with the addition of each new member to the data sharing group. And if you choose to copy indexes, this table space can grow even more.

Consider increasing the frequency with which you remove rows from this table space, or increasing the size. To see the definition used for SYSLGRNX, see installation job DSNTIJIN.

To increase the space allocation for SYSLGRNX, use access method services:

1. Stop the table space.
2. Rename the existing SYSLGRNX data set.
3. Define a larger SYSLGRNX data set with the original name.
4. Using only DSN1COPY, copy the contents of the renamed data set into the new SYSLGRNX data set.
5. Restart the table space.

**Application design planning**

This section includes information relevant to programmers that develop applications for use in DB2 data sharing environments.

**Using the NO CACHE option when creating sequences**

Sequences provide unique, sequential, numeric values to DB2 applications. Applications can use sequence numbers for a variety of purposes, including the avoidance of concurrency and performance problems that can result when applications generate their own sequence numbers. Unlike application-generated sequences, DB2 sequences allow multiple transactions to concurrently increment the sequence number, and DB2 guarantees that each sequential value is unique. DB2 does not wait for a transaction that has incremented a sequence to commit before allowing the sequence to be incremented by another transaction. No retained locks are held to prevent access to the sequence.

The CACHE option of the CREATE SEQUENCE statement directs DB2 to preallocate a specified number of sequential values in memory. This performance and tuning option provides faster access to the sequence by eliminating synchronous I/O to the SYSEIBM.SYSEQUENCES table each time an application requests a new sequence number.

DB2 always generates sequence numbers in order of request. However, because sequences can be shared across multiple members of a data sharing group, each member must have its own cache of unique consecutive numbers for the same sequence. Therefore, in situations where transactions from different members are requesting the next sequence number from the same sequence, values are not assigned in strict numeric order.

**Example**: Assume that members DB2A and DB2B share a sequence named SEQ1 that starts with 1, increments by 1, and caches 20 values at a time. DB2A might initially get the cache of values 1 to 20, and DB2B might initially get values 21 to 40. If the transaction that is associated with DB2A makes the first request for a sequence number, a value of 1 is provided to the application. If the transaction that is associated with DB2B makes the next request for a sequence number, a value of 21 is provided. Assuming that sequence number requests continue to arrive from the transactions that are associated with members DB2A and DB2B in this manner (one from DB2A and then one from DB2B), the values 2, 22, 3, 23, and so on, are provided.

**Important**: In data sharing environments, if sequence numbers must be provided in strict numeric order, use the NO CACHE option of the CREATE SEQUENCE statement. For detailed information about creating sequences by using the CREATE SEQUENCE statement, see [DB2 SQL Reference](#).

**Applications using CICSPlex System Manager**

CICSPlex® System Manager (CICSPlex SM) is a system-management tool that lets you manage several Customer Information Control System (CICS) systems as if they were one. The dynamic transaction routing program supplied with CICSPlex SM balances the enterprise workload dynamically across the available application owning regions. CICSPlex SM lets you manage a variable workload without
operator intervention and maintains consistent response times. It can do this because it routes transactions away from busy regions and from those that are failing or likely to fail, which improves throughput and conceals problems from end users.

**Be aware of the storm drain effect**

In some situations, your DB2 applications must be sensitive to a "resource-unavailable" condition. For example, assume that a database is stopped for planned maintenance, and that the application receives SQLCODE -904 from the system and ends normally. If your application ignores the SQLCODE -904 message, CICSPlex SM might continue to route work to the stopped system because it appears to complete its work rapidly. This is sometimes called the *storm drain effect*.

When both of the following conditions are true, the storm drain effect can occur:

- The CICS attachment facility is down.
- You are using INQUIRE EXITPROGRAM to avoid AEY9 abends.

Again, because there has not been an abend, it appears as if work completes rapidly at that subsystem.

**Writing a CICS exit to avoid the storm drain effect**

The information in this section, up to "Migrating transactions that have ordered dependencies" on page 67, is Product-sensitive Programming Interface and Associated Guidance Information about CICS and DB2.

You can write a resource manager interface program exit routine, XRMIOUT, to avoid the storm drain effect caused by SQLCODE -904 (resource unavailable). This exit routine does not avoid the storm drain problem caused by using INQUIRE EXITPROGRAM to avoid AEY9 abends.

Using XRMIOUT, you can intercept the return from the resource manager. The exit routine can determine whether:

- The resource manager is DB2.
- SQLCODE -904 is in the SQL communication area (SQLCA).

If these conditions exist, abend the transaction instead of ending the transaction normally.

To determine if DB2 is the resource manager, compare ‘DSNCSQL’ with the value stored at the address that is included with the UEPTRUEN parameter that is passed to XRMIOUT, as shown in Figure 18 on page 66.
To find the SQLCODE:

1. Find UEPTRUEP in the DFHUEPAR parameter list that is passed to XRMIOUT. UEPTRUEP contains the address of the DFHUEPAR parameter list that is passed to DSNCSQL.

2. Find UEPHMSA in DSNCSQL's DFHUEPAR parameter list. UEPHMSA points to the register save area that contains the application's registers.

3. Find register 1 in the register save area. Register 1 contains the address of the application parameters.

   The DSNXRDI macro maps the application parameters that are passed by the precompiler to DB2. The mapping macro is contained in the data set library prefix.SDSNMACS.

4. Find RDICODEP in the DSNXRDI structure. RDICODEP contains the address of the SQL communication area (SQLCA).

5. Find the SQLCODE. The SQLCODE offset is documented in Appendix D of *DB2 SQL Reference*.

For more information about the XRMIOUT exit, see *CICS Transaction Server for z/OS Customization Guide*.

**Use a CICS enhancement to avoid the storm drain effect**

The CICS Transaction Server for z/OS helps you avoid the storm drain effect. With the CICS Transaction Server for z/OS, you do not need to use XRMIOUT to check for resource unavailable conditions.

CICS Transaction Server for z/OS also lets you benefit from the INQUIRE EXITPROGRAM command without causing the storm drain effect. For more information about the INQUIRE EXITPROGRAM command, see *CICS Transaction Server for z/OS System Programming Reference*. 
Migrating transactions that have ordered dependencies

This section pertains only to those limited cases in which one transaction, called an originating transaction, updates DB2 using INSERT, UPDATE, or DELETE. Then, before completing phase 1 of the commit process, the originating transaction spawns a second transaction that is dependent on the updates performed by the first transaction. These types of transactions are referred to as order-dependent transactions.

Description of the problem

In some situations, a dependent transaction can encounter a “row not found” condition that does not occur in non-data sharing environments.

Nevertheless, even in a non-data sharing environment, dependent transactions need to tolerate the “row not found” condition in cases where the originating transactions roll back.

When the problem might occur

The “row not found” condition can occur if all of the following conditions are true:

- The originating transaction spawns a dependent transaction before it completes phase 1 of the commit process.
- The dependent transaction runs on a different member than the member on which the originating transaction runs.

Because each member has its own local buffer pools, the uncommitted buffered pages that are updated by the originating transaction are not immediately visible to the dependent transaction.

- The dependent transaction attempts to access a row that was updated by the originating transaction.
- The dependent transaction is not bound with an isolation level of repeatable read.

Preventing the problem

To prevent these periodic “row not found” situations, consider using the IMMEDWRITE(YES) option of BIND/REBIND for a plan or package that spawns dependent transactions that might run on other members. Using the IMMEDIATE WRITE subsystem parameter also works. IMMEDWRITE(YES) means that DB2 writes the page as soon as the buffer update completes. DB2 writes the data to one of the following structures or devices:

- The group buffer pool
- Disk storage for GBPCACHE NO group buffer pools
- Disk storage for GBPCACHE NONE page sets
- Disk storage for GBPCACHE SYSTEM page sets

The IMMEDIATE WRITE parameter on installation panel DSNTIP4 can override the value of the IMMEDWRITE bind option with which a plan or package is executed on a data sharing member. For information on which values of the parameter cause the bind option to be overridden, see “Other recommendations” on page 73.

The following alternatives can help solve the order-dependent transaction problem:

- Ensure that the originating transaction does not schedule the dependent transaction until the originating transaction has completed phase 1 of commit.
- Run the dependent transaction with an isolation level of repeatable read.
• If the dependent transaction is currently running with an isolation level of cursor stability AND CURRENTDATA(NO), changing it to use CURRENTDATA(YES) can sometimes solve the problem.

• Add statement retry logic to handle the return of a “row not found” condition.

• Run the dependent transaction on the same member as the originating transaction.

Table 10 illustrates the implied hierarchy when using the IMMEDWRI subsystem parameter and the IMMEDWRITE option of the BIND and REBIND commands.

<table>
<thead>
<tr>
<th>IMMEDWRITE bind option</th>
<th>IMMEDWRI subsystem parameter</th>
<th>Value at run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

Note:

YES always has precedence whether it is the subsystem parameter or the bind option. Updated pages that are group buffer pool-dependent are written at, or before, phase 1 of the commit process.

**Binding plans and packages if you are moving to a new machine**

You do not need to rebind a plan or package to run it on a data sharing group. However, if you are moving to a new machine that has different performance characteristics (moving from a System/390® microprocessor to a z/Series microprocessor cluster, for example), it is to your advantage to rebind plans and packages on the machine on which they will be running. See “Access path selection in a data sharing group” on page 272 for more information.
Chapter 3. Installing and enabling DB2 data sharing

This chapter provides an overview of how to make a data sharing group. Table 11 points you to the procedures you need to create or migrate to a data sharing group. The complete set of installation panels and steps are shown in DB2 Installation Guide.

Be sure to read “Choosing parameters for members” for guidance on choosing specific subsystem parameters.

Table 11. Data sharing options

<table>
<thead>
<tr>
<th>If you have this...</th>
<th>And you want this...</th>
<th>Read this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>No system</td>
<td>Version 8 data sharing</td>
<td>“Installing a new data sharing group” on page 77.</td>
</tr>
<tr>
<td>A Version 8 non-sharing DB2 subsystem</td>
<td>The originating member of a data sharing group</td>
<td>“Enabling DB2 data sharing” on page 80.</td>
</tr>
<tr>
<td>One member in the group</td>
<td>More members in the group</td>
<td>“Adding new members” on page 81.</td>
</tr>
<tr>
<td>Separate DB2 subsystems</td>
<td>Merged DB2 subsystems into a single group</td>
<td>“Merging existing DB2 data into the group” on page 83.</td>
</tr>
<tr>
<td>A Version 7 data sharing group</td>
<td>A Version 8 data sharing group</td>
<td>“Migrating an existing data sharing group to a new release” on page 91.</td>
</tr>
</tbody>
</table>

The following topics provide additional information:

- “Creating a data sharing group” on page 75
- “Renaming a member” on page 78
- “Testing the data sharing group” on page 89
- “Updating subsystem parameters for a member” on page 91
- “Falling back and remigrating” on page 100
- “Disabling and re-enabling DB2 data sharing” on page 102
- “Removing members from the data sharing group” on page 105

Choosing parameters for members

Every member of a data sharing group must have its own unique load module for member parameters (sometimes called DSNZPARM in a non-data sharing environment).

The load module for member parameters is built by job DSNTIJUZ and stored in the prefix.SDSNEXIT target library. Every member must use a different name for its parameters’ load module because the prefix.SDSNEXIT target library can be shared among all members of the data sharing group. The installation process requires that you provide the name of the load module for a member.
Recommendation: Name each member’s load module using the convention DSNZPxxx, where xxx includes the number in the member name and the group identifier. For example, DB1A’s subsystem parameters load module could be named DSNZP01A.

The subsystem parameters’ load module name for a member is an optional parameter on the EXEC statement in the JCL procedure used to start the ssnmMSTR address space. This optional parameter provides support for an operator (or automated operations) not specifying the subsystem parameter load module name when starting a member. The following is the format for specifying the parameter on the EXEC statement of the ssnmMSTR JCL procedure:

//IEFPROC EXEC PGM=DSNYASCP,PARM='ZPARM(DSNZPxxx)',...

The scope and uniqueness of DB2 subsystem parameters

Even though various parameters affect the operation of only a single member, some parameters must have the same value on all the members. For example, each catalog alias name must be the same.

The values of other parameters must be unique for each member. For example, each member specifies a different BSDS name.

The values of most parameters do not need to be unique. Recommendations are offered for some of these parameters. In the following tables, the parameter is indicated by the installation panel parameter name. However, some of the parameters do not reside in the DSNZPxxx load module.

Parameter values that must be different on each member

Table 12 shows the values of the installation panel parameters that must be different on every member of a data sharing group. You specify values for these parameters when a member is installed.

Table 12. Parameters that must have different values on each member of a data sharing group

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Panel ID</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Logs: COPY 1 PREFIX</td>
<td>DSNTIPH</td>
<td>Each member writes to its own recovery log.</td>
</tr>
<tr>
<td>Active Logs: COPY 2 PREFIX</td>
<td>DSNTIPH</td>
<td>Each member writes to its own recovery log.</td>
</tr>
<tr>
<td>Archive Logs: COPY 1 PREFIX</td>
<td>DSNTIPH</td>
<td>Each member writes to its own recovery log.</td>
</tr>
<tr>
<td>Archive Logs: COPY 2 PREFIX</td>
<td>DSNTIPH</td>
<td>Each member writes to its own recovery log.</td>
</tr>
<tr>
<td>Bootstrap Data Sets (BSDS): COPY 1 NAME</td>
<td>DSNTIPH</td>
<td>Each member has its own BSDS.</td>
</tr>
<tr>
<td>Bootstrap Data Sets (BSDS): COPY 2 NAME</td>
<td>DSNTIPH</td>
<td>Each member has its own BSDS.</td>
</tr>
<tr>
<td>COMMAND PREFIX</td>
<td>DSNTIPM</td>
<td>The command prefix used to route commands to this member.</td>
</tr>
</tbody>
</table>
Table 12. Parameters that must have different values on each member of a data sharing group (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Panel ID</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 NETWORK LUNAME</td>
<td>DSNTIPR</td>
<td>Even if you do not use distributed database, this identifier is required to ensure that logical unit of work IDs (LUWIDs) are unique across the members of a data sharing group.</td>
</tr>
<tr>
<td>DB2 PROC NAME</td>
<td>DSNTIPX</td>
<td>JCL procedure for the DB2-established stored procedures address space.</td>
</tr>
<tr>
<td>MEMBER IDENTIFIER</td>
<td>DSNTIPJ</td>
<td>The unique identifier for this IRLM.</td>
</tr>
<tr>
<td>MEMBER NAME</td>
<td>DSNTIPK</td>
<td>The name for this member.</td>
</tr>
<tr>
<td>PARAMETER MODULE</td>
<td>DSNTIPO</td>
<td>The name of the parameter load module for this member.</td>
</tr>
<tr>
<td>PROCNAME</td>
<td>DSNTIPI</td>
<td>The name of the IRLM procedure that z/OS invokes if IRLMAUT=YES.</td>
</tr>
<tr>
<td>RESYNC PORT</td>
<td>DSNTIP5</td>
<td>The port used for resynchronization of two-phase commit processes when using TCP/IP network protocols.</td>
</tr>
<tr>
<td>SUBSYSTEM NAME</td>
<td>DSNTIPM</td>
<td>The DB2 subsystem identifier for this member.</td>
</tr>
<tr>
<td>SUBSYSTEM SEQUENCE</td>
<td>DSNTIPM</td>
<td>The DB2 subsystem sequence number for this member.</td>
</tr>
<tr>
<td>SUBSYSTEM NAME (IRLM)</td>
<td>DSNTIPI</td>
<td>The name of the IRLM subsystem associated with a particular member. This name must be unique within the Parallel Sysplex.</td>
</tr>
<tr>
<td>WORK FILE DB</td>
<td>DSNTIPK</td>
<td>The name of the work file database for this member.</td>
</tr>
</tbody>
</table>

Parameter values that must be the same on every member

Table 13 shows the installation panel parameters that must have the same values for every member of the data sharing group.

Table 13. Parameters that must have the same values on every member of a data sharing group

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Panel ID</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTH AT HOP SITE</td>
<td>DSNTIP5</td>
<td>The site at a second server (hop’ site) whose authorization is to be checked.</td>
</tr>
<tr>
<td>CATALOG ALIAS</td>
<td>DSNTI PA2</td>
<td>The DB2 catalog alias name.</td>
</tr>
<tr>
<td>DBADM CREATE AUTH</td>
<td>DSNTI PP</td>
<td>Whether an authorization ID with DBADM authority can create a view for another authorization ID on that database’s tables. Or if the authorization ID can create an alias for itself or another authorization ID for a table in that database.</td>
</tr>
</tbody>
</table>
Table 13. Parameters that must have the same values on every member of a data sharing group (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Panel ID</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE PROTOCOL</td>
<td>DSNTIP5</td>
<td>The default protocol to use when an application program contains three-part names.</td>
</tr>
<tr>
<td>DB2 LOCATION NAME</td>
<td>DSNTIPR</td>
<td>The location name for the entire data sharing group. This name is required.</td>
</tr>
<tr>
<td>DRDA PORT</td>
<td>DSNTIP5</td>
<td>The DRDA port number for the entire data sharing group. This name is required if using TCP/IP network connections.</td>
</tr>
<tr>
<td>EXTENDED SECURITY</td>
<td>DSNTIPR</td>
<td>Determines the content of error codes returned to a client when a connection request fails because of security errors. Also allows users to change their passwords if their host passwords expire.</td>
</tr>
<tr>
<td>GROUP ATTACH</td>
<td>DSNTIPK</td>
<td>The group attachment name that allows programs to generically attach to any member of the group. The default is the member name of the first installed member of the group.</td>
</tr>
<tr>
<td>GROUP NAME</td>
<td>DSNTIPK</td>
<td>The DB2 group name.</td>
</tr>
<tr>
<td>IRLM XCF GROUP NAME</td>
<td>DSNTIPJ</td>
<td>The IRLM group name.</td>
</tr>
<tr>
<td>INSTALL DD CONTROL SUPT. (and related parameters)</td>
<td>DSNTIPZ</td>
<td>All members of the data sharing group must use the same set of data definition control registration tables, or unpredictable results can occur.</td>
</tr>
<tr>
<td>MINIMUM DIVIDE SCALE</td>
<td>DSNTIPF</td>
<td>Whether to retain at least three digits to the right of the decimal point after any decimal division.</td>
</tr>
<tr>
<td>SITE TYPE</td>
<td>DSNTIPO</td>
<td>All members on the remote site, after they need to become the local site, must have the same value (LOCALSITE).</td>
</tr>
<tr>
<td>STATISTICS HISTORY</td>
<td>DSNTIPO</td>
<td>Which inserts and updates are recorded in catalog history tables.</td>
</tr>
<tr>
<td>SYSTEM ADMIN 1 SYSTEM ADMIN 2 SYSTEM OPERATOR 1 SYSTEM OPERATOR 2</td>
<td>DSNTIPP</td>
<td>Installation SYSADM and SYSOPR authorities.</td>
</tr>
<tr>
<td>TCP/IP ALREADY VERIFIED</td>
<td>DSNTIP5</td>
<td>Whether incoming TCP/IP requests are accepted by DB2 without a password or RACF PassTicket. This option must be the same on all members or requesters will have inconsistent results.</td>
</tr>
<tr>
<td>TRACKER SITE</td>
<td>DSNTIPO</td>
<td>Whether DB2 is a tracker site for disaster recovery services.</td>
</tr>
<tr>
<td>VARCHAR FROM INDEX</td>
<td>DSNTIP4</td>
<td>Whether the VARCHAR column is retrieved from the index.</td>
</tr>
</tbody>
</table>
Other recommendations

Table 14 shows the installation panel parameters that can have the same or different values for members of a data sharing group.

Table 14. Recommended parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Panel ID</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEALLOC PERIOD</td>
<td>DSNTIPA</td>
<td>The length of time during which an archive read tape unit is allowed to remain unused before it is deallocated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Recommendation:</strong> Do not archive to tape. If you must, however, you should specify 0 for this parameter unless you intend to run all RECOVER jobs from the same member. Specifying a deallocation delay means that the tape is not available to any other member until the deallocation time expires.</td>
</tr>
<tr>
<td>DEFAULT BUFFER POOL FOR USER DATA</td>
<td>DSNTIP1</td>
<td>The default buffer pool for table spaces that are created without a specified default. For consistent results when creating objects on different members, make this default the same on all members.</td>
</tr>
<tr>
<td>DEFAULT BUFFER POOL FOR USER INDEXES</td>
<td>DSNTIP1</td>
<td>The default buffer pool for indexes that are created without a specified default. For consistent results when creating objects on different members, make this default the same on all members.</td>
</tr>
<tr>
<td>DEVICE TYPE 1</td>
<td>DSNTIPA</td>
<td>The device type or unit name for storing archive logs.</td>
</tr>
<tr>
<td>DEVICE TYPE 2</td>
<td></td>
<td><strong>Recommendation:</strong> Archive the first copy of the log to disk.</td>
</tr>
<tr>
<td>EDMPOOL STORAGE SIZE</td>
<td>DSNTIPC</td>
<td>The size of the environmental descriptor manager (EDM) pool in kilobytes. Whatever value is calculated from the installation CLIST, consider adding more storage because of the way data sharing updates database descriptors in the EDM pool.</td>
</tr>
</tbody>
</table>
### Table 14. Recommended parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Panel ID</th>
<th>Comment</th>
</tr>
</thead>
</table>
| IMMEDIATE WRITE       | DSNTIP4  | Determines the IMMEDWRITE bind option used for plans and packages that are run on the member:  
  NO   The IMMEDWRITE bind option that is specified for the plan or package is used. The default is NO.  
  YES  The IMMEDWRITE(YES) option is used for all plans and packages. This option might impact performance.  
  The IMMEDWRITE bind option controls when DB2 writes updated group buffer pool dependent pages to the coupling facility (or to disk for GBPCACHE NONE | SYSTEM objects). DB2 can either write the GBP-dependent pages immediately, before phase 1 commit, or at phase 1 commit. For more information about the bind option, see [DB2 Command Reference](#). For information about using the bind option to avoid "row not found" conditions for transactions that have ordered dependencies, see "Migrating transactions that have ordered dependencies" on page 67. |
| READ COPY2 ARCHIVE    | DSNTIPO  | Whether the second copy of the archive log should be read first for restart and recovery.  
  **Recommendation:** This option should have the same value (either YES or NO) for all members of the group. Otherwise, the parameter, as it is set on the member that owns the archive log data set, determines which copy is used. |
| READ TAPE UNITS       | DSNTIPA  | The maximum number of dedicated tape units that can be allocated to read archive log tape volumes concurrently.  
  **Recommendation:** Do not archive to tape. If you must, however, it is vital that you have enough tape units allocated to the member doing the recovery to merge the archive logs from all members in the group that have updated the object being recovered. Thus, if there are eight members in the group, make sure you specify at least eight on this panel for each member. See "The impact of archiving logs in a data sharing group" on page 159 for more information about archiving to tape. |
| RECORDING MAX         | DSNTIPA  | The maximum number of archive logs to be recorded in the BSDS.  
  **Recommendation:** All members should use the maximum value of 1000, making it easier to transfer a workload from one member to another. |
### Table 14. Recommended parameters (continued)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Panel ID</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETAINED LOCK TIMEOUT</td>
<td>DSNTIPI</td>
<td>A multiplier to apply to the timeout value for a connection when that connection is waiting for an incompatible lock held by another failed member. Locks that are held by failed members are called retained locks. <strong>Recommendation:</strong> If you have automatic restart, or some other restart automation that quickly restarts failed members, choose a non-zero value for this parameter.</td>
</tr>
<tr>
<td>START IRLM CTRACE</td>
<td>DSNTIPI (or TRACE=YES on IRLM startup procedure)</td>
<td>Whether diagnostic traces are activated when IRLM is started. <strong>Recommendation:</strong> Diagnostic traces should be activated for all members of the group. The negligible performance overhead is outweighed by the benefit of easier problem resolution. Only those systems that have reached the limits of processor capacity should consider not automatically activating these traces.</td>
</tr>
</tbody>
</table>

### DSNHDECP parameters

There is a single DSNHDECP load module for each release of DB2 in a data sharing group. The application programming defaults contained within the DSNHDECP load module are considered global for the group. The load module is created during the installation of the DB2 group. It cannot be modified during the installation of a member.

The DSNHDECP parameter DECPSSID has a special meaning in a data sharing group. DECPSSID contains the group attachment member name for the data sharing group. This allows utilities, TSO attachment, RRSAF, and CAF applications to attach to any member in the group.

### Creating a data sharing group

To create a data sharing group, add one member at a time. The first member (the originating member) can either be created as a new installation or enabled for data sharing from an existing Version 8 member.

Before enabling DB2 data sharing, read the following topics:
- “Recommended approach for moving to DB2 data sharing” on page 76
- “Sharing DB2 libraries” on page 76
- “Ensuring that installation jobs access the correct JCL procedures” on page 77
- “Establishing system affinity for installation jobs” on page 77

### Recommended approach for moving to DB2 data sharing

Moving to DB2 data sharing is a big step. Plan this move carefully because after you enable data sharing, it is very difficult to disable it. Disabling should only be considered if your long-term plans are to disable data sharing.
Before enabling data sharing, test other major new functions in the release on a single system, and then build and try a test data sharing group. When you are ready to begin a move to production, you must avoid having to fall back to the previous release.

**Build and try a test data sharing group**

Here is one approach to testing your data sharing group:

1. Install Version 8 as a single system, and test it with dummy data, or copies of production data, in order to test some of the new functions. Run old applications and begin new application development work.
   
   In the meantime, prepare the hardware and define the coupling facility structures to enable data sharing.

2. Enable data sharing on the test system.

   You have a single-system data sharing group at this point, and this can help you find any initial configuration problems. Make sure that old applications work in this environment.

3. Install additional members in the test group.

   Run applications from different members in the group to fully exercise the group buffer pools and cross-system locking.

**Move to production**

When you are ready to move to production:

1. Migrate your existing Version 7 member to Version 8, but do not use any of the new function yet.

   Alternatively, you can enable data sharing on Version 7 and then migrate to Version 8, but it is not recommended.

2. Start to use new functions when you are sure the release is stable and you will not need to fall back.

3. Tune applications to contain the level of locking and lock contention rates.

   To reduce the effects of locking contention in a data sharing environment, first control locking costs in a non-data sharing environment. This gives you a baseline from which to perform further tuning after the move to data sharing. See Part 5 (Volume 2) of *DB2 Administration Guide* for information about reducing locking contention. See "Improving concurrency" on page 219 for information about reducing contention in a data sharing environment.

4. Enable data sharing on the originating member, and run applications on this one-member data sharing group.

5. Install additional members as needed.

**Sharing DB2 libraries**

DB2 target and distribution library data sets can be shared among all members of a data sharing group. There is no need for each member of the group to have its own set of target and distribution libraries. Sharing these libraries reduces the effort of installing and maintaining the different members of the data sharing group. As described in "Administering a database" on page 15, sharing libraries can also help ensure that members are using the same exit routines. And sharing libraries also simplifies the tasks of defining a new member to a data sharing group. The member installation process supports sharing of the libraries.
Ensuring that installation jobs access the correct JCL procedures

If you have more than one procedure library, ensure that the installation jobs access the right set of procedures by using a JCLLIB statement to specify the order in which procedure libraries are searched.

The JCLLIB statement looks like this:

```
//ddname JCLLIB ORDER=(library[,library...])
```

The JCLLIB statement must follow the JOB statement and precede the first EXEC statement in the job. You can have DB2 insert this statement in your JCL for you by entering it on installation panel DSNTIPY.

For more information on the JCLLIB statement, see z/OS MVS JCL Reference.

Establishing system affinity for installation jobs

You must ensure that the installation jobs run on the z/OS system on which the appropriate member is running. There are several z/OS installation-specific ways to make sure that this happens. These include:

- For JES2 multi-access spool (MAS) systems, use the following JCL statement:

  ```
  /*JOBPARM
  SYSAFF=cccc
  ```

  Where `cccc` is the JES2 name. You can specify an asterisk (SYSAFF=*) to indicate that the job should run on the system from which it was submitted.

- For JES3 systems, use the following JCL statement:

  ```
  /*MAIN SYSTEM=(main-name)
  ```

  The `main-name` is the JES3 name.

z/OS MVS JCL Reference describes the preceding JCL statements. You can edit the jobs manually, or you can enter the preceding statements on installation panel DSNTIPY and have DB2 insert these statements for you.

Your installation might have other mechanisms for controlling where batch jobs run, such as by using job classes.

Installing a new data sharing group

This procedure describes how to install and immediately enable DB2 data sharing on a new Version 8 member.

**Use this procedure only in low-risk situations.** The recommended approach is to migrate to, or install, a Version 8 member, use it for a while, and then enable data sharing as described in “Enabling DB2 data sharing” on page 80.

However, if you decide to install and immediately enable data sharing on a new Version 8 member, this member becomes the originating member of the data sharing group. This member’s DB2 catalog is used as the DB2 catalog for the data sharing group.

To install the new data sharing group:

1. On installation panel DSNTIPA1, specify:

   ```
   INSTALL TYPE ===> INSTALL
   DATA SHARING ===> YES
   ```
2. On installation panel DSNTIPP1, specify 1 to indicate the group data sharing function.

3. On installation panel DSNTIPK, specify:

GROUP NAME  ==> group name
MEMBER NAME  ==> originating member name

Verify that the originating member name is unique within the z/OS Parallel Sysplex. Installation job DSNTIJMV edits the ssnmMSTR startup procedure with the group name and member name you specify here.

4. Complete the installation panels, specifying parameters according to the guidelines in "The scope and uniqueness of DB2 subsystem parameters" on page 70.

5. Complete all installation steps, as described in Part 2 of DB2 Installation Guide.

6. Run the installation verification procedures (IVP), as described in Part 2 of DB2 Installation Guide.

Renaming a member

Renaming a member is an activity that you should plan for very carefully. Because every installation has a different configuration, this procedure is not guaranteed to be complete. If you are interested in changing the high level qualifier for data sets, see the procedure in Part 2 (Volume 1) of DB2 Administration Guide.

If you want to rename a member of the data sharing group, you should do so either before enabling DB2 data sharing or during the process of enabling DB2 data sharing. Because DB2 must be shut down during the enabling process, you can perform the tasks necessary to rename the member at the same time.

This section is divided into two parts:

- "Tasks that require an IPL"
- "Tasks at enable time" on page 79

In the following example procedure, the member name for DB2P is changed to DB1A to conform to the naming convention for data sharing that is used in this publication.

**Tasks that require an IPL**

If you choose to do any of the following tasks, you must IPL z/OS to pick up the changes:

- Modify the IEFSSNxx member for this subsystem to include the group attachment name. To avoid having to modify JCL for all your jobs, use the existing subsystem name as the group attachment name. This step is necessary to ensure that there are no problems when the call attachment and TSO attachment facilities try to use the group attachment name.

**Example:** If the existing IEFSSNxx member looks like:

DB2P,DSN3INI,'DSN3EPX,?'

Change it to look like:

DB2P,DSN3INI,'DSN3EPX,?,S, DB2P'

- Add new IEFSSNxx definitions for the new name and prefix:

  DB1A,DSN3INI,'DSN3EPX,-DB1A,S, DB2P'

  DJ1A

  Optionally, you can use the z/OS command SETSSI to add the new IEFSSNxx statements without an IPL. You can add the new statements when you add the
definitions for the new names or later during the enabling process. Do not forget to add these names to the IEFSSNxx member before you IPL again.

The command prefix -DB1A is not a subset or a superset of DB2P’s command prefix. For example, ?DB1A is invalid in this context.

- Make sure that RACF definitions are in place to handle the new subsystem names. Add the correct names to the following tables:
  - The RACF router table
  - The started procedures table (ICHRIN03), if used

**Tasks at enable time**

You can do the following tasks when you bring down the DB2 subsystem for the enabling procedure described in "Enabling DB2 data sharing” on page 80.

1. Define the correct profile names for the DSNR class.
2. Replicate existing PERMIT commands to allow users and groups to access the new profiles.
3. Complete the installation panels for enabling data sharing. (You cannot change the subsystem name on the installation panels.)

   **On installation panel DSNTIPH, modify the archive prefix names to include the member name, like DB2PCAT.DB1A.ARCGLG1.** Specify the old subsystem name (DB2P) as the group attachment name.
4. Stop DB2 with MODE(QUIESC).
5. Run job DSNTIJUZ to assemble and link-edit the new subsystem parameter data set (DSNZP:xx) and DSNHDEC.
6. Rename the BSDS and active log data sets with the new prefix, like DB2PCAT.DB1A.BSDS01.
7. Update the BSDS with the renamed log data sets. **Include the same RBA ranges as the original active log data sets:**
   a. Run the utility print log map (DSNJU004) to obtain the start and end RBAs.
   b. Use access method services to rename the log data set.
   c. Run the utility change log inventory (DSNJU003) to delete the active logs with the old names.
   d. Run the utility change log inventory (DSNJU003) to add the renamed active logs with the correct ranges. You do not need to add the archive log data sets, because the archive log data sets you specified on panel DSNTIPH will replace them.
8. If necessary, increase the size of the BSDS, as described in “Increasing the size of the BSDS” on page 63.
9. If necessary, increase the size of the SYSLGRNX table space, as described in “Increasing the size of the SYSLGRNX table space” on page 63.
10. Rename the startup procedures. For example, change DB2PMSTR to DB1AMSTR. Do not forget to change the BSDS names to the new names in the ssmMSTR startup procedure.
11. Make sure that CICS can connect to the new subsystem. There are several ways of doing this, depending on which level of CICS you are running. Here are a couple of techniques:
   - Change the CICS DB2CONN to use the new DB2 subsystem name, reassemble, and link-edit.
   - Modify the JCL for CICS to include the new subsystem name on the INITPARM.
12. Make sure that IMS can connect to the new subsystem.
13. Enter the command START DB2 and continue with the rest of the enabling process.

Enabling DB2 data sharing

After you migrate to, and test, Version 8, enable the originating member for data sharing. The enabling process simply allows this existing subsystem to be the originating member of a data sharing group; it does not allow you to change the subsystem name. (See "Renaming a member" on page 78 for information about how to rename a subsystem.) This originating member’s catalog is used as the DB2 catalog for the data sharing group.

The enabling CLIST tailors these jobs:

<table>
<thead>
<tr>
<th>DSNTEJ0</th>
<th>DSNTEJ2U</th>
<th>DSNTEJ6S</th>
<th>DSNTESD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNTEJ1</td>
<td>DSNTEJ3C</td>
<td>DSNTEJ6T</td>
<td>DSNTESE</td>
</tr>
<tr>
<td>DSNTEJ1L</td>
<td>DSNTEJ3P</td>
<td>DSNTEJ6U</td>
<td>DSNTIJDE</td>
</tr>
<tr>
<td>DSNTEJ1P</td>
<td>DSNTEJ4C</td>
<td>DSNTEJ61</td>
<td>DSNTIJFT</td>
</tr>
<tr>
<td>DSNTEJ1S</td>
<td>DSNTEJ4P</td>
<td>DSNTEJ62</td>
<td>DSNTIJGF</td>
</tr>
<tr>
<td>DSNTEJ1T</td>
<td>DSNTEJ5A</td>
<td>DSNTEJ7</td>
<td>DSNTIJIN</td>
</tr>
<tr>
<td>DSNTEJ2A</td>
<td>DSNTEJ5C</td>
<td>DSNTEJ71</td>
<td>DSNTIJMV</td>
</tr>
<tr>
<td>DSNTEJ2C</td>
<td>DSNTEJ5P</td>
<td>DSNTEJ73</td>
<td>DSNTIJTM</td>
</tr>
<tr>
<td>DSNTEJ2D</td>
<td>DSNTEJ6</td>
<td>DSNTEJ75</td>
<td>DSNTIJVC</td>
</tr>
<tr>
<td>DSNTEJ2F</td>
<td>DSNTEJ6D</td>
<td>DSNTESA</td>
<td>DSNTIJUZ</td>
</tr>
<tr>
<td>DSNTEJ2P</td>
<td>DSNTEJ6P</td>
<td>DSNTESC</td>
<td></td>
</tr>
</tbody>
</table>

The enabling CLIST also edits the DB2I CLISTs.

Procedure: To enable data sharing:
1. On installation panel DSNTIPA1, specify:
   
   INSTALL TYPE ==> INSTALL
   DATA SHARING ==> YES

2. On installation panel DSNTIPP1, specify 3 to enable data sharing.
3. On installation panel DSNTIPK, specify:
   
   GROUP NAME ==> group name
   MEMBER NAME ==> originating member name

   Verify that the originating member name is unique within the z/OS Parallel Sysplex. Installation job DSNTIJMV edits the ssnmMSTR startup procedure with the group name and member name you specify here.

4. Complete the installation panels, specifying parameters according to the guidelines in "The scope and uniqueness of DB2 subsystem parameters" on page 70.
5. Complete the following installation steps, as described in Part 2 of DB2 Installation Guide:
   a. Stop DB2 activity.
      See Migration Step 9 in Part 2 of DB2 Installation Guide for a detailed list of steps about how to stop activity.
   b. Installation Step 1: Define DB2 to z/OS: DSNTIJMV.
   c. Installation Step 3: Define system data sets: DSNTIJIN.
For the enable process, DSNTIJIN only alters the current active log data sets to use SHAREOPTIONS (2 3).

d. Installation Step 5: Define DB2 initialization parameters: DSNTIJUZ.
e. Installation Step 7: Record DB2 data to SMF (optional).
f. Installation Step 9: Connect DB2 to TSO: DSNTIJVC.
   This step is necessary only if you are using the group attachment name.
g. Installation Step 12: IPL z/OS.
   This step is only necessary if you are changing the command prefix, or adding or changing the group attachment name.
h. Installation Step 13: Start the DB2 subsystem.
i. Installation Step 14: Define temporary work files: DSNTIJTM.
j. Installation Step 18: Image copy the DB2 directory and catalog: DSNTIJIC (optional).
   If you decide to image copy the DB2 directory and catalog, use the DSNTIJIC job that is generated during the installation or migration of the originating member.
   You do not need to make an image copy for the DB2 catalog or user data sets for recovery because DB2 uses image copies made before you enabled data sharing.
k. Installation Step 19: Verify the installation.

6. Optionally, run the installation verification procedures (IVP), or a subset of these, as described in Part 2 of [DB2 Installation Guide].

If you ran the complete set of IVP sample jobs after you migrated to Version 8, you should not need to run these jobs again. When you start the originating member, DB2 checks your coupling facility, group, and member definitions and verifies that data sharing is enabled. You can also verify that the group has been correctly established by issuing the DB2 command DISPLAY GROUP after the originating member has completed startup.

Adding new members

Always add a member to a data sharing group as a new installation. After being added to the group, the new member begins using the DB2 catalog of the originating member. (See z/OS Parallel Sysplex Overview: An Introduction to Data Sharing and Parallelism for information on creating a Parallel Sysplex.)

When adding a new member to a group, you might need to make changes during installation to allow more XCF groups, or more XCF members per group. Or, you might need to ‘‘widen’’ the locks in the IRLM lock structure (for example, if the structure was initially allocated with a maximum number of seven members, and the eighth member is joining the group).

DB2 does not have an automatic way to ‘‘merge’’ catalogs and resolve naming conflicts. If you have applications that are currently running on several existing DB2 subsystems, your migration plan might include procedures for moving the relevant data and applications from those DB2 subsystems onto one or more of the group members. Your migration plan should also include procedures for resolving any catalog naming conflicts that result. See ‘‘Merging existing DB2 data into the group’’ on page 83 for more information about this.

After you install a new data sharing group or enable an existing DB2 subsystem for data sharing, you can add new data sharing members.
Jobs that the add-member CLIST tailors:

<table>
<thead>
<tr>
<th>DSNTIJIN</th>
<th>DSNTIJMV</th>
<th>DSNTIJD</th>
<th>DSNTIJDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNTIJTM</td>
<td>DSNTIJUZ</td>
<td>DSNTIJFT</td>
<td>DSNTIJGF</td>
</tr>
</tbody>
</table>

**Procedure:** To add a new member to a data sharing group:

1. On installation panel DSNTIPA1, specify:

   INSTALL TYPE ===> INSTALL
   DATA SHARING ===> YES
   .
   INPUT MEMBER NAME ===> originating member's output PDS member
   OUTPUT MEMBER NAME ===> new member's output PDS member

2. On installation panel DSNTIPP1, specify 2 for adding a member.

3. On installation panel DSNTIPK, specify a new member name:

   MEMBER NAME ===> new member name

   Verify that the **new member name** is unique within the z/OS Parallel Sysplex.
   Installation job DSNTIJMV edits the ssmnMSTR startup procedure with the member name you specify here.

4. Complete the installation panels, specifying parameters according to the guidelines in "The scope and uniqueness of DB2 subsystem parameters" on page 70.

   **Recommendation:** Rename the tailored SDSNSAMP data set for each member.
   This data set contains tailored JCL for each member, including jobs that are used for disabling and re-enabling data sharing, if that should ever become necessary. If you do not rename the SDSNSAMP data set, it is overwritten when you install a new member. Rename the data set by choosing a new name for the prefix.NEW.SDSNSAMP data set on installation panel DSNTIPT. For example, use prefix.member_name.SDSNSAMP.

5. Complete the following installation steps, as described in Part 2 of DB2 Installation Guide:

   a. Installation Step 1: Define DB2 to z/OS: DSNTIJMV.
   b. Installation Step 3: Define system data sets: DSNTIJIN.

      When you add a new member, DSNTIJIN does not define catalog and directory data sets. It does define the new BSDS and active log data sets to use SHAREOPTIONS (2 3).

   c. Installation Step 4: Define DB2 initialization parameters: DSNTIJUZ.
   d. Installation Step 5: Initialize system data sets: DSNTIJD.

      When you add a new member, DSNTIJD does not initialize the catalog and directory data sets. It does add the new active log data sets to the BSDS.

   e. Installation Step 7: Record DB2 data to SMF (optional).
   f. Installation Step 8: Establish subsystem security (optional).
   g. Installation Step 10: Connect IMS to DB2 (optional).
   h. Installation Step 11: Connect CICS to DB2 (optional).
   i. Installation Step 12: IPL z/OS.

      You can use the z/OS command SETSSI to dynamically add the new DB2 and IRLM subsystems without having to IPL z/OS. For example, the following two commands can be used to add subsystems for DB2 and IRLM to MVS3:

      RO MVS3,SETSSI ADD,SUB=DB3A,INITRTN=DSN3INI,INITPARM='DSN3EPX,-DB3A,S,DB0A'
      RO MVS3,SETSSI ADD,SUB=DJ3A
**Recommendation:** Add DB2 to the IEFSSN.xx member so that it can be used on a subsequent IPL. For more information, see z/OS MVS Using the Subsystem Interface for more information.

j. Installation Step 13: Start the DB2 subsystem.
   Transactions might fail because work files are not defined until Installation Step 15.

k. Installation Step 15: Define temporary work files: DSNTIJTM.
   You need to create a backup copy of the DB2 directory and catalog. Use the DSNTIJIC job generated during the installation or migration of the originating member to do this.

m. Installation Step 20: Verify the installation.

6. Test the data sharing group, as described in "Testing the data sharing group" on page 89.

---

**Merging existing DB2 data into the group**

DB2 provides no automatic way to merge existing DB2 data or existing DB2 subsystems into a data sharing group. Read "Deciding if merging is the right thing to do" on page 59 before attempting to merge subsystems into a single data sharing group.

**Merging subsystems**

If you want to merge existing subsystems into a single data sharing group, follow these steps:

1. Choose one of the subsystem to be the originating member.
2. Move data and data definitions from the other DB2 subsystems to the originating member.
3. Add those other DB2 subsystems to the group using the new member installation process.

**Merging data**

If you have an application that is currently running on independent DB2 subsystems, you might decide that it is an application that will work well in a data sharing group. In that case, you must move the data and merge the catalog definitions for that data from the independent DB2 subsystems into the data sharing group. Because the DB2 subsystems still exist, you cannot reuse their subsystem names when installing the subsystems as new members into the data sharing group.

DB2 does not provide an automated way to move catalog definitions from an independent DB2 subsystem into the catalog of the data sharing group. If you have procedures and tools in place now to move applications from test to production, or to handle merging databases from enterprise reorganizations or mergers, those same procedures can be used to move applications into the data sharing group.

**Existing distributed applications**

If you move existing data to the data sharing group, it is likely that the location name of objects will change. Existing distributed applications that remotely reference an object by its three-part name must be changed to the new name. And any aliases on that table must also be dropped and recreated with the new location name.

Any application containing explicit SQL CONNECT statements that reference an old location name must be modified. Any DB2 plan that uses an old location name for the CURRENTSERVER keyword must be bound again.

**Procedure for moving data**

In the following procedure the term “target” refers to the DB2 subsystem that you are moving data to (the data sharing group), and “source” refers to the DB2 you are moving data from.

1. Decide the method for moving data. Three options are outlined below; you can use a combination of the specified methods. Each of the methods assumes that the objects that are to be created in the target system are created with the exact DDL that is used in the source system. Descriptions of methods 1, 2, and 3 are referred to throughout the procedure, and are described in detail below.

   **Method 1: Unload or Reload**: The data that is to be moved is unloaded from the source DB2 subsystem and loaded into the target data sharing group by using the DB2 online utilities, UNLOAD or LOAD. This approach is the most simple because tables are created without concern for object IDs, and data is unloaded or reloaded using conventional techniques. The major disadvantage of this method is that it is the slowest of the 3 methods, in terms of performance. The UNLOAD and RELOAD methods operate against rows of a table, whereas the other techniques operate against pages or data sets as their level of granularity. Additionally, this method requires two to three times the disk space that the other methods require, depending on how you unload or reload (source, target and intermediate unload).

   **Method 2: DSN1COPY with OBID translation**: The data that is to be moved is copied from the source system to the target system using the DB2 offline utility DSN1COPY. DSN1COPY is usually faster than unload or reload because it moves pages of data, rather than rows. This method is more complicated because the object IDs in the source data will need to be translated. For more information about OBID translation with DSN1COPY, refer to the [DB2 Utility Guide and Reference](#).

   **Method 3: Use DB2 VSAM data sets from the source system in the target system**: The VSAM data sets that are used to hold the table space and index space data from the source system are used in the target system, without copying any data. Because this method does not require copying the data, it is the fastest of the three methods, in terms of performance. It is, however, the most complicated. Another advantage of this method is that it requires less disk space than the other methods; this method is generally used for very large table or index spaces.

2. Choose a catalog that belongs to one of the DB2 subsystems to be the “original” catalog for the data sharing group. This DB2 subsystem is considered the originating member of the group.

   You should consider many different factors when choosing which DB2 subsystem to be the originating member of the group. For example, it makes sense to choose the member with the most database objects as the originating member in order to minimize the number of objects that you move.

   However, if all DB2 subsystems are mostly equivalent, and you are not planning to use DSN1COPY, consider the log RBA values of the existing subsystems. Compare the end-of-log RBAs with the high-order 6-bytes of the time-of-day clock timestamp on each of their systems (this is called the truncated timestamp). The usual case is that the RBA is less than the truncated timestamp. In this case, any DB2 subsystem is chosen as the originating member.
In the event that the current end-of-log RBA in any of the existing DB2 subsystems is higher than the 6-byte truncated timestamp value at the time you are ready to enable sharing, you have the following choices:

- Choose the DB2 subsystem that has the highest RBA as the originating member.
- Use DSN1COPY with the RESET option to reset the log RBAs in each data and index page to 0 when you move databases from other DB2 subsystems to the data sharing group.

3. Resolve name conflicts among the objects and authorization IDs in the data sharing group.

4. Create the objects on the target subsystem.

When creating objects while you are using methods 1 or 2 to move the data, the CREATE statements can be entered in any order. DB2 assigns new OBIDs for these objects. If you are using method 2, consult the [DB2 Utility Guide and Reference](#) for information about obtaining object IDs that are needed for OBID translation with DSN1COPY.

If you are using method 3, you must perform the following steps:

a. Consider creating the data sets with the minimum size settings to save space. The actual target system VSAM data sets created by the DDL execution are not used because the VSAM data sets that are from the source system are used instead. By choosing to create the objects with minimum sizes, you eventually need to issue ALTER TABLESPACE and ALTER INDEX statements to change the PRIQTY and SECQTY after the CREATE. If, however, you are using the same high-level qualifier for the source and target systems (which is not recommended), you must create the objects with DEFINE NO, and take additional steps later. DEFINE NO is needed when keeping the same high-level qualifier, because the VSAM data set name would be identical in the source and target systems. In this case, DB2 would not be able to define the data set on the target system during the create processing because the data set already exists.

b. Query the SYSIBM.SYSTABLES in the DB2 catalog on the source subsystem to get the table OBID for tables that are within the databases that are being moved. Additionally for both the source and target systems, the DBID and PSID are needed for table spaces (from SYSIBM.SYSTABLESPACE) and the DBID and ISOBID are needed for indexes (from SYSIBM.SYSINDEXES). These will be used to build REPAIR jobs to modify these IDs in the header page, and potentially the first space map page in a later step.

c. Use the OBID clause on the CREATE TABLE statement on the target subsystem to specify an OBID that is the same as the table OBID on the source subsystem. Verify that the OBID that you specify is available. If an OBID is being used for another object within the same database, such as an index or referential constraint, DB2 does not allow you to create the table with the specified OBID.

**Recommendation:** To help guarantee the availability of OBIDs for all tables within a database, defer the creation of all indexes and referential constraints until all tables are created. All CREATE TABLE statements must have the OBID clause to guarantee that they are assigned the correct OBID. If an explicit table space name is specified, then the CREATE TABLESPACE statement must come immediately before the first CREATE TABLE statement for that table space.

This helps prevent a group of CREATE TABLESPACE statements from using up OBIDs that are needed for the tables.
For methods two or three, the objects in the target system will be created with several assumptions that might require the additional steps that are outlined below:

- The VSAM data sets for each new object are created using the ‘I’ prefix for each of the data sets. It is possible that on the source system online, REORG has executed, causing the data sets to have a ‘J’ prefix. If this is the case, you will need to code the DSN1COPY for method two, to have the correct ‘J’ data set designation or rename the data set back to the ‘I’ prefix for method three in a later step. You should query the source catalog for column IPREFIX in SYSIBM.SYSTABLEPART for table spaces, and SYSIBM.SYSINDEXPART for indexes, to find any ‘J’ prefix objects.
- If any of the index names involved in the move are greater than eight characters in length, DB2 generates a unique INDEXSPACE name to be used in naming the VSAM data set. The INDEXSPACE name generated will most likely be different in the source and target systems. You should query the source and target catalogs to get a list of the INDEXSPACE names for the indexes that have a name greater than eight characters (found in SYSIBM.SYSINDEXES). These will be used in a later step.

5. On the source DB2 subsystem, run the REORG utility on any table spaces for which the following conditions are both true for a table in the table space:
   - A column has been added using the ALTER TABLE ADD COLUMN utility with no subsequent REORG.
   - The columns of a table in the table space are all fixed-length.
   If you are unsure if a table meets this criteria, query SYSIBM.SYSTABLES for those tables in which CREATEDTS does not equal ALTEREDTS. However, there is no way to tell from the DB2 catalog whether the ALTER consisted of adding a column. By comparing the CREATEDTS and ALTEREDTS, you will at least gather a list of candidates. If all rows have CREATEDTS equal to ALTEREDTS, running the REORG utility is unnecessary.

6. Move the data choosing one of the three methods that match the ones listed above:
   - Method 1: Unload / Reload:
     a. Start the objects on the source system in RO mode.
     b. Execute the UNLOAD utility on the source system to unload the data to a sequential file.
     c. Execute the LOAD utility on the target system to load the data that was unloaded when performing the step listed above.
   - Method 2: DSN1COPY with OBID translation:
     a. Stop the object, for both table space and index, on the source and target systems.
     b. Execute DSN1COPY with OBID translation and the RESET option to copy the data from the source system to the target system. If the online REORG utility has been run, be sure to code the proper I/J data set names in the DSN1COPY job. Also, if any indexes were created with names greater than eight characters long you will need to incorporate the INDEXSPACE name differences retrieved in an earlier step.
     c. Start the objects on the target systems R/W.
   - Method 3: Use DB2 VSAM data sets from source system in the target system:
     a. Stop the table spaces and index spaces in the target system.
     b. If the high level qualifier (HLQ) is not changing between the source and target systems, which is not recommended, then perform the following
steps on the target system. Note that this process assumes that the objects were created in the target system with DEFINE NO utility.

1) Change SPRMCTU in the DSN6SPRM macro to one. This enables the catalog to be updated. Also, change DLFREQ to zero, which disables the down level detection. You must note the original value, because you will change DLFREQ back to the original value in a later step.
   Assemble the changed zparm and restart the target system to pick up the change.

2) Using SQL, update the SPACE column from SYSEMs.SYSTABLEPART and SYSEMs.SYSINDEXPART from -1 to 0 for the table and index spaces that you are moving using method three.

3) Run the REPAIR DBD REBUILD utility for the databases created earlier. This step is needed because the above step changed the SPACE value in the DB2 catalog from -1 to 0, but the value is also in the DBD. This means that when the REBUILD utility is run, it will take the information from the catalog and rebuild the DBD in the directory, making DB2 think that the data set has already been defined. NOTE: Do not try to access the tables at this point. Both systems will think that the data set is theirs, but only one system will actually be able to access the data set. The REPAIR DBD REBUILD utility only should be executed after all table spaces and index spaces that are being moved by method three for a particular database, have their catalogs updated via SQL.

4) Change SPRMCTU back to zero, and then DLFREQ to the previous value it was assigned. Next, reassemble the zparm and recycle the target system so it recognizes the change.
   The identifiers consist of two, 2-byte fields, HPGDBID and HPGPSID. You must locate and replace these identifiers with the new DBID and PSID of the target system as follows:
   - For non-partitioned page sets, locate and replace the 4-byte fields starting at X'0C', which begins on page zero.
   - For partitioned page sets, locate and replace the 4-byte fields starting at X'0C' on page zero for each partition. Refer to the description of the REPAIR utility in Part 2 of DB2 Utility Guide and Reference for more information on how to specify partition numbers.

5) If the table space that you are working with is segmented and compressed, and a dictionary exists, you must also run the REPAIR utility on the source system to change the OBID in the first space map page. See 87 for more information on running this utility.

6) Stop the object, for table space and index, on the source system.

7) If the HLQs of the source and target systems are different, you must run the IDCAMS ALTER statements. Running the IDCAMS ALTER statements changes the HLQ of the VSAM data sets from the source system to be the new HLQ of the target system. If the HLQ is changing, then the VSAM datasets of the target system must be first deleted before running IDCAMS ALTER. Additionally, if you know from completing one of the previous steps that the online REORG utility has been executed on the source system, make sure that you change any ‘i’ data sets to match the target system. The target system should have all ‘i’ data sets, as they were newly created. Also, if any indexes were created with names greater than eight characters long, you will need to incorporate the INDEXSPACE name differences retrieved in an earlier step.

8) Start the table spaces and index spaces in the target system.
9) Use the REPAIR utility with the LEVELID option to reset the level indicator of the page sets to a neutral value on the target system.

c. Stop the object, for table space and index, on the source system.

d. You now have the option to drop the database objects on the source subsystem. If you do not choose to drop the objects at this point, and used method three to move data while keeping the same HLQ for the data sets, then you should consider altering the VCAT name in the source system to a different invalid value. You should alter the VCAT name so that the source DB2 does not try to open the data set that is now in use by the target system.

e. Make full image copies of all data. This is the earliest time at which data recovery can occur after the merge.

f. Run the RUNSTATS utility on the target system.

g. Bind all plans and packages on the target subsystem that were bound on the source subsystem. Bind any plans and packages on the target subsystem that have changed because of name conflict resolution. Grant the appropriate authorizations to all plans and packages.

Running REPAIR to modify OBID for a compression dictionary

When using method three to move data, if the table space is segmented and compressed, then there are additional invocations of the REPAIR utility that may need to be run. If the segmented compressed table space has a dictionary, the space map page that contains the dictionary has an OBID that is the same as the table space that was modified in the header page in one of the above steps. The following step can be used to modify the OBID in the dictionary to match the new value in the header page. The following is an example from DSN1PRNT with the FORMAT option. On the header page, or page zero, you will see something similar to the output displayed in [Figure 19].

Figure 19. Sample header page of DSN1PRNT command with the FORMAT option.

The following figure shows the first space map page, or page 1:

Figure 20. Sample output from first space map page of DSN1PRNT command with FORMAT option.
and 6, following the segment number. The OBID in these four segments needs to be modified with the REPAIR utility to match the new OBID in the header page. The remaining segments are for the table and do not need to be modified. You can see this in Figure 20 on page 88, which shows that the table OBID is X'0003'.

If you run DSN1PRNT on pages 0 and 1, and you do not see the same OBID in the space map page as the one listed on the header page, it means that the dictionary is not been built yet. If the dictionary is not built, you only see the table segments (segments X'5' thru X'E' shown above). If this is the case, you can omit this current step.

Figure 20 on page 88 shows that the number of segments that need to be modified is dependent on the SEGSIZE value of the table space. Table 15 lists the offset of the OBIDs that need to be modified.

Table 15. The offset of OBIDs that need to be modified in order of SEGSIZE

<table>
<thead>
<tr>
<th>SEGSIZE</th>
<th>Offset of OBIDs to REPAIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>X'20'</td>
</tr>
<tr>
<td></td>
<td>X'29'</td>
</tr>
<tr>
<td></td>
<td>X'32'</td>
</tr>
<tr>
<td></td>
<td>X'3B'</td>
</tr>
<tr>
<td>8</td>
<td>X'20'</td>
</tr>
<tr>
<td></td>
<td>X'2B'</td>
</tr>
<tr>
<td>12</td>
<td>X'20'</td>
</tr>
<tr>
<td></td>
<td>X'2D'</td>
</tr>
<tr>
<td>16–64</td>
<td>X'20'</td>
</tr>
</tbody>
</table>

Testing the data sharing group

When you installed DB2, sample objects were created in job DSNTJEJ1. The DSNTESD member of prefix.SDSNSAMP contains SQL statements that refer to these objects. These SQL statements can be used to test group buffer pool caching, global lock serialization, and concurrency in the data sharing group. Perform these tests after installing several data sharing members.

Testing group buffer pool caching

Use the SQL statements in DSNTESD to verify that the group buffer pool operates correctly.

1. Run SPUFI on more than one data sharing member, using DSNTESD as the input data set. Specify AUTOCOMMIT=YES on the SPUFI panel.

   Run SPUFI on the different members serially, a few seconds apart if possible. (The runs must be close enough together to avoid having DB2 close the page set because of infrequent updates. The default amount of time between updates before DB2 switches the page set from read-write to read-only is 10 minutes, as specified on the RO SWITCH TIME parameter of installation panel DSNTIPN. RO SWITCH CHKPTS can also cause the data set to switch to read-only. It is a number of consecutive checkpoints.

   By running the SQL statements serially, DB2 detects inter-DB2 read/write interest on the table space and index and uses the group buffer pool.)
2. Verify that ITEM_COUNT increases by 5 after each run.

3. Issue the following command to determine whether the table space and index are using the group buffer pool:
   
   ```
   DIS DB(DSN8D81A) SPACENAM(DSN8S81S,XPARTS) LOCKS
   ```

   If the P-lock state is IX or SIX, then the table space and index are group buffer pool dependent, which is the correct state.

4. Issue the following command to display the statistics for GBP0:
   
   ```
   DIS GBP00L(GBP0) GDETAIL
   ```

   In the group detail statistics, look for non-zero values in the READS and WRITES values of the display. This indicates that DB2 is using the group buffer pool successfully for caching.

**Testing global lock serialization**

Use the SQL statements in DSNTESD to verify that locks are acquired and released correctly across multiple data sharing members.

1. Run SPUFI, using DSNTESD as input, on member 1. Specify AUTOCOMMIT=NO.

   Because you have inserted data into DSN8810.PARTS but have not committed, member 1 holds global locks.

2. Run SPUFI, using DSNTESD as input, on member 2. Specify AUTOCOMMIT=NO.

   Because member 1 holds global locks, member 2 must wait to perform the insert.

3. In less than one minute, commit on member 1. (If you wait too long to commit, member 2 will experience a lock timeout.)

   The global locks should be released, and member 2 should be able to proceed. Verify that ITEM_COUNT has increased by five.

**Testing concurrency**

Use the SQL statements in DSNTESD to test concurrency within the data sharing group.

1. Run SPUFI concurrently on different data sharing members. Specify AUTOCOMMIT=YES.

   Global locking ensures that inserts to DSN8810.PARTS are coordinated across data sharing members.

2. Verify that ITEM_COUNT increases by five each time the run completes successfully.

**Testing Sysplex query parallelism**

There is no sample procedure to test Sysplex query parallelism, but you can use your own data to confirm that a single query can be processed on more than one member of the data sharing group. Choose an existing query that you know uses CP parallelism, such as a SELECT COUNT(*) for a table in a large partitioned table space, and use the following procedure to force DB2 to split the query across multiple members of the data sharing group:

1. Decide which member will be the query coordinator for your test, and make sure that the COORDINATOR parameter on installation panel DSNTIPK is set to YES for that member.

2. Make sure that all potential assistants have the ASSISTANT parameter on installation panel DSNTIPK set to YES.
3. Make sure that the query does not include one of the restrictions listed in Part 5 (Volume 2) of [DB2 Administration Guide].

4. Run EXPLAIN on the query.
   An X in the PARALLELISM_MODE column of the PLAN_TABLE output indicates that this query can be split across multiple members.

5. Set buffer pool allocation thresholds on the members that you want considered as possible assistants:
   ALTER BUFFERPOOL (BPn) VPSIZE(z) VPSEQT(100) VPXSEQT(100) VPXSEQT(100)
   Ensure that the VPSIZE is large enough to support parallel processing. Start with at least 50 buffers on each query assistant.

6. Make sure that accounting trace class 3 is active on the parallelism coordinator.

7. Run the query.

8. Inspect the IFCID 0003 trace record. Field QWA01RBN corresponds to the number of assisting members. This value should be greater than zero.

---

**Updating subsystem parameters for a member**

There are no group-wide subsystem parameters to update. To update subsystem parameters for a member of a data sharing group, specify the following installation option:

```
INSTALL TYPE ===> UPDATE
```

This option updates parameters for the member.

---

**Migrating an existing data sharing group to a new release**

**Important:** Migration to DB2 Version 8 is only supported from Version 7.

Before beginning the migration process, confirm that Version 7 is at the proper service level. Also, check your coupling facilities to ensure that the appropriate service levels are installed. Having the wrong service levels installed can result in data corruption.

- If you have coupling facilities at CFLEVEL=7, then you need service level 1.06 or above.
- If you have coupling facilities at CFLEVEL=8, then you need service level 1.03 or above.

No specific service level requirements exist for CFLEVELs other than 7 and 8. Use the z/OS D CF command to display the service levels for IBM coupling facilities.

Migrating an existing data sharing group requires careful planning:

1. Read the information about migration considerations in Part 2 of [DB2 Installation Guide]

2. Read the information in “Mixed releases in a data sharing group” on page 92.

3. Make a plan to migrate the data sharing group in as short a time period as possible.

4. Ensure that maintenance through the Version 8 fallback SPE is applied to all active members.

   You must apply the fallback SPE (and its prerequisite APARs) before attempting to migrate any member of the group. Stop and restart each member to activate the change.
Version 8 members cannot start if any one of the active Version 7 members does not have the SPE applied. Similarly, if a Version 8 member is started, a Version 7 member cannot start unless it has the fallback and coexistence SPE applied.

To prepare for fallback, keep the subsystem parameter load module used by Version 7.

5. Follow the procedure in “Procedure to migrate the data sharing group” on page 99. Refer to Part 2 of DB2 Installation Guide for detailed information about migration.

During the migration to Version 8, other group members can be active, but they might experience delays or timeouts if they attempt to access catalog objects that are being updated or that are locked by migration processing.

# Mixed releases in a data sharing group

DB2 allows the data sharing group to remain available while you migrate members of the group to the newest release. However, planning the migration for periods of low activity in the group is recommended because, due to locks that are obtained by the catalog migration utility (CATMAINT), parts of the DB2 catalog are unavailable during the migration of the first member. The parts that are unavailable vary from release to release, depending on what catalog parts require modification.

The purpose of coexistence is to allow applications to continue to access DB2 data while you migrate the members of the data sharing group. However, you must weigh the benefit of continuous availability against the operational costs of running in coexistence mode: many new functions are not available, and there are some system management issues to consider.

If you do not require continuous availability, consider shutting down the group during the migration to avoid the coexistence environment. If you need to run in coexistence mode, make a plan to migrate the members in as short a time as possible so that you can minimize the operational complexity.

DB2 allows a maximum of two different release levels to coexist in a data sharing group at any one time.

Determining the release of the data sharing group

When the first data sharing member starts Version 8, the entire group is considered to be at that level, even though not all members of the group have migrated. This means that the group level is Version 8. You can display the group level by issuing the DISPLAY GROUP command, as shown in Figure 21 on page 93.
Determining the function level of the IRLM group

IRLM communicates coexistence information by using function levels. A function level is an ever-increasing number that each IRLM can use to tell other IRLMs in the group what level of function it supports. The group function level is the minimum of the individual IRLM function levels for all IRLMs that can coexist. An IRLM that tries to join a data sharing group is prevented from doing so by active members that cannot coexist with the new IRLM’s function level.

When the function level for the group changes, that change is serialized by IRLM with lock structure rebuilds. In most cases, however, the lock structure does not actually do a full rebuild. The first phase of the rebuild is enough to quiesce the work and cause the function level change to occur. These “partial” rebuilds take place when an IRLM joins or leaves the group and if that activity causes the group function level to change. For example, if the IRLM group is currently at function level n, and the IRLM member that wants to join the group is at n-1, the partial rebuild occurs to lower the group function level. Conversely, if the lowest level member leaves the group, the partial rebuild might occur if the group can coexist at a higher function level.

To determine IRLM function levels, enter the following command:
MODIFY irmlproc,STATUS,ALLI

Example: Assume a data sharing group exists with two members running DB2

Version 8 compatibility mode and one member running DB2 Version 7. Issuing the

MODIFY irmlproc,STATUS,ALLI command from one of the two Version 8 members

produces output that is similar to the following output:

```
DXR1031 DJ1A001 STATUS
IRLMs PARTICIPATING IN DATA SHARING GROUP FUNCTION LEVEL=1.022
IRLM NAME IRLMID STATUS LEVEL SERVICE MIN LEVEL MIN SERVICE
   DJ1A  001  UP  2.022  HIR2220  2.022  HIR2220
   DJ2A  002  UP  2.022  HIR2220  2.022  HIR2220
   DJ3A  003  UP  1.022  PQ52360  1.012  PN90337
```

The IRLMs in the preceding example are at group function level 1.022, which is the lowest level of any of the individual members (DJ3A). The MIN_LEVEL field
shows the minimum level with which this IRLM can coexist. MIN_SERVICE indicates the service or release that corresponds with that MIN_LEVEL.

Example: Assume the same data sharing group as described in the previous example. Issuing the MODIFY irlmproc, STATUS,ALLI command from the Version 7 member produces output that is similar to the following output:

```
  DXR103I DJ3A003 STATUS
  IRLMS PARTICIPATING IN DATA SHARING GROUP FUNCTION LEVEL=022
  IRLM_NAME IRLMID STATUS LEVEL SERVICE MIN_LEVEL MIN_SERVICE
  DJ1A 001 UP 094 HIR2220 022 HIR2220
  DJ2A 002 UP 094 HIR2220 022 HIR2220
  DJ3A 003 UP 022 P052360 012 PN90337
  DXR103I End of display
```

Call attachment and TSO attachment coexistence

While operating in a coexistence environment, you can attach to either release of DB2 with your existing TSO logon procedures or with JCL. After you migrate all members of the group to DB2 Version 8, update those procedures and jobs to point to the Version 8 load libraries.

Avoiding automatic rebinds

When developing your migration plan, keep in mind that new functions introduced in Version 8 are not available to members of the group who have not yet migrated. Thus, it is best to either:

- Migrate all members to the new release before attempting to use new utilities, commands, or options. Do not allow members to run any applications that include new SQL function until all members have migrated to the new release.

- Prevent packages and plans that are bound on Version 8 from executing on members that have not yet been migrated. Or, do not bind plans and packages on Version 8.

This prevention serves two purposes. First, if plans or packages that are bound on Version 8 use new functions, you can avoid the application errors that would occur if the plan or package tried to execute an SQL statement that is not allowed in Version 7. Second, you can prevent the automatic rebind that occurs when any plan or package that is bound on Version 8 is run on Version 7. It also prevents the automatic rebind that occurs when a Version 8-bound plan or package that was automatically rebound on Version 7 is later run on Version 8. (Such a bind is called a remigration rebind.)

If enforcing where a plan or package runs is not possible, consider how you want to handle binds and automatic rebinds while two releases are coexisting. One approach is to disallow all binds and disable all automatic rebinds on Version 8 members. The other approach is to disable only remigration rebinds. By disabling remigration rebinds, you avoid the thrashing that can occur when a plan or package is rebound each time it runs on a member of a different version.

Disallowing all binds: To avoid automatic rebinds on a Version 8 member:

- Specify NO for the AUTO BIND parameter of installation panel DSNTIPO. This disables all automatic rebinds on the Version 8 member for any reason. Therefore, you cannot run a plan or package on a Version 8 member if it has gone through the following scenario:
  1. Binds on a Version 8 member.
  2. Runs on a Version 7 member.
     (This causes an automatic rebind on the Version 7 member.)
  3. Attempts to run on a Version 8 member.
(This returns a -908 SQLCODE (SQLSTATE '23510') because DB2 must automatically rebind the plan or package on Version 8 before running it on the Version 8 member.)

- Use the resource limit facility to disallow BIND operations. Do this by inserting rows in the resource limit specification table (RLST) to set RLFFUNC to "1" and RLFBIND to "N". This ensures that nobody binds plans or packages on Version 8.

**Example:** The following is an INSERT statement for the RLST that disallows all BIND operations for all authorization IDs (except those with installation SYSADM or installation SYSOPR authority) for all packages and plans:

```
INSERT INTO authid.DSNRLSTxx
(RLFFUNC,RLFBIND) VALUES('1','N');
```

After all the members of a data sharing group have migrated to the current release, enable automatic rebinds by setting AUTO BIND=YES, and allow bind operations by changing the RLST accordingly or by stopping the resource limit facility using the STOP RLIMIT command.

**Disallowing only the automatic remigration rebind:** To avoid the automatic remigration rebind situation, specify COEXIST for the AUTOBIND parameter on installation panel DSNTIPO of the Version 8 members. Automatic rebind then occurs on Version 8 in the following circumstances:

- The plan or package is marked invalid.
- You migrate to a future release, bind a plan or package on that release, and then run the plan or package on Version 8.

When all members are at Version 8, you do not need to change the COEXIST value. The behavior is the same as if you had specified AUTOBIND YES.

**Recommendations for BIND**

If DSN is at Version 8 and the member that is named in the DSN command is at Version 7, using certain bind options causes a BIND or REBIND subcommand to be rejected.

If you are migrating from Version 7, the following bind option causes rejection:

- **ENCODING** for BIND and REBIND PLAN or PACKAGE

To avoid problems, make sure that the member named in the DSN subcommand matches the load libraries that are used for the DSN command.

**Recommendations for utilities**

Until all members of the data sharing group are running Version 8, avoid using any of the new utility functions available in Version 8. However, as long as you use utility options that are supported in Version 7, utilities can attach to either a Version 7 or Version 8 member.

The utilities batch program (DSNUTILB) is split into multiple load modules: a release-independent load module called DSNUTILB, multiple release-dependent module DSNUT710 or DSNUT810, and multiple utility-dependent load modules. The utility-dependent load modules are listed in Appendix E of DB2 Utility Guide and Reference. To operate in a mixed-release data sharing environment, you must have DSNUT610 (if applicable), DSNUT710 (if applicable), and DSNUT810. And you must have all utility-dependent load modules and their aliases for the utilities that you have purchased available to the utility jobs that operate across the data sharing group. This is shown in "Running purchased utilities in coexistence mode" on page 96.
See "Changing STEPLIB in DSNUPROC" and "Cross-copy into load libraries" below for instructions on making these load modules available.

Changing STEPLIB in DSNUPROC: The recommended way of making the release-dependent modules available for utility jobs is to change the STEPLIB in DSNUPROC to include the other release, as in the following example:

```
//DSNUPROC
PROC
LIB='DSN810.SDSNLOAD',
// SYSTEM=DSN,
// SIZE=0K,UID='',UTPROC=''
//DSNUPROC EXEC PGM=DSNUTILB,REGION=&SIZE,
// PARM='&SYSTEM,&UID,&UTPROC'
//STEPLIB DD DSN=&LIB,DISP=SHR;
// DD DSN=DSN710.SDSNLOAD,DISP=SHR coexistence
//SYSPRINT DD SYSOUT=* 
//UPRINT DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
///DSNUPROC PEND REMOVE *, FOR USE AS INSTREAM PROCEDURE 
```

Cross-copy into load libraries: Another approach, not recommended for long-term use, is to cross-copy the release-dependent modules into the load libraries of the other release. For example, copy DSNUT710 into the Version 8 load libraries, and copy DSNUT810 and all applicable utility-dependent load modules into the Version 7 load libraries. The problem with this approach is that you must repeat this procedure every time you apply maintenance to these modules. Thus, as with coexistence in general, this approach is only for short-term use.

Example: The following shows sample JCL to perform the cross-copy:

```
// CROSCOPY PROC D710TPRE='DSN810',
// D610TPRE='DSN610',
// RGN=4096K,SOUT=''
//* ********** FOR EXECUTION OF IEBCOPY - DB2 POST-INSTALLATION ***
//* **************************************************************************
//COPY EXEC PGM=IEBCOPY,REGION=&RGN
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//* ********** DB2 TARGET LIBRARIES ***************************************
//* /D710LOAD DD DSN=&D710TPRE..SDSNLOAD,DISP=OLD 
//D810LOAD DD DSN=&D810TPRE..SDSNLOAD,DISP=OLD
//* 
// COEXIST EXEC PROC=DSNTIJCO
//SYSIN DD *
COPYMOD INDD=(((D81LOAD,R)),OUTDD=D71LOAD 
SELECT MEMBER=(DSNUT810) -- add any utility-dependent modules here
COPYMOD INDD=(((D71LOAD,R)),OUTDD=D81LOAD 
SELECT MEMBER=(DSNUT710)
```

Running purchased utilities in coexistence mode

In Version 8, each utility that you purchase has a separate load module that is associated with one of the 20 aliases shown in Table 16. When a utility is executed, it is loaded by using its alias. By including in the link list the aliases for those utilities you want, you can run in coexistence mode without specifying a STEPLIB in the JCL.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATMAINT</td>
<td>DSNU81AA</td>
</tr>
</tbody>
</table>
Table 16. Version 8 utility aliases (continued)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK</td>
<td>DSNU81AB</td>
</tr>
<tr>
<td>COPY</td>
<td>DSNU81AC</td>
</tr>
<tr>
<td>DIAGNOSE</td>
<td>DSNU81AD</td>
</tr>
<tr>
<td>LISTDEF</td>
<td>DSNU81AE</td>
</tr>
<tr>
<td>LOAD</td>
<td>DSNU81AF</td>
</tr>
<tr>
<td>MERGECOPY</td>
<td>DSNU81AG</td>
</tr>
<tr>
<td>MODIFY</td>
<td>DSNU81AH</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>DSNU81AI</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>DSNU81AJ</td>
</tr>
<tr>
<td>REBUILD</td>
<td>DSNU81AK</td>
</tr>
<tr>
<td>RECOVER</td>
<td>DSNU871AL</td>
</tr>
<tr>
<td>REORG</td>
<td>DSNU81AM</td>
</tr>
<tr>
<td>REPAIR</td>
<td>DSNU81AN</td>
</tr>
<tr>
<td>REPORT</td>
<td>DSNU81AO</td>
</tr>
<tr>
<td>RUNSTATS</td>
<td>DSNU81AP</td>
</tr>
<tr>
<td>STOSPACE</td>
<td>DSNU81AQ</td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>DSNU81AR</td>
</tr>
<tr>
<td>UNLOAD</td>
<td>DSNU81AS</td>
</tr>
<tr>
<td>COPYTOCOPY</td>
<td>DSNU81AT</td>
</tr>
</tbody>
</table>

**Note:**

1. Use this utility only under the direction of IBM Software Support.

The last character in the alias is a utility identifier, which is used to associate the alias with the utility name in the feature. Aliases are used because some utilities are shipped in multiple features, and the names differ according to the feature’s naming convention. Although the names differ, the utilities are identical. For example, the COPY utility that is shipped with the OPERATIONAL (JDB771K) feature is named DSNU8OLC, and the COPY utility that is shipped with the RECOVERY AND DIAGNOSTIC (JDB771M) feature is named DSNU8RLC. DSNU8OLC and DSNU8RLC are identical except for their names. Notice that the last character in both names (DSNU8OLC and DSNU8RLC) is a C. In Table 16 on page 96, you can see that the alias for the COPY utility also ends with a C (DSNU81AC).

Table 17 on page 98 and Table 19 on page 98 show the new load module names for Version 8. Compare the last character in each module name with the aliases in Table 16 on page 96 to associate that module with a feature.

Table 17. Core utility load modules that are shipped with the base

<table>
<thead>
<tr>
<th>DSNU8CLA</th>
<th>DSNU8CLD</th>
<th>DSN8CLE</th>
<th>DSNU8CLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNU8CLJ</td>
<td>DSNU8CLN</td>
<td>DSNU8CLO</td>
<td>DSNU8CLR</td>
</tr>
</tbody>
</table>

The characters ‘8CL’ in each module name indicate Version 8, core feature, load module.
Table 18. Load modules that are shipped with the OPERATIONAL (JDB771K) feature

<table>
<thead>
<tr>
<th>Module</th>
<th>Module</th>
<th>Module</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNU8OLC</td>
<td>DSNU8OLF</td>
<td>DSNU8OLK</td>
<td>DSNU8OLL</td>
</tr>
<tr>
<td>DSNU8OLM</td>
<td>DSNU8OLP</td>
<td>DSNU8OLQ</td>
<td>DSNU8OLS</td>
</tr>
</tbody>
</table>

The characters '8OL' in each module name indicate Version 8, operational feature, load module.

Table 19. Load modules that are shipped with the RECOVERY AND DIAGNOSTIC (JDB771M) feature

<table>
<thead>
<tr>
<th>Module</th>
<th>Module</th>
<th>Module</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNU8RLB</td>
<td>DSNU8RLC</td>
<td>DSNU8RLG</td>
<td>DSNU8RLH</td>
</tr>
<tr>
<td>DSNU8RLK</td>
<td>DSNU8RLL</td>
<td>DSNU8RLT</td>
<td></td>
</tr>
</tbody>
</table>

The characters '8RL' in each module name indicate Version 8, recovery feature, load module.

Recommendation for group restart

If a group restart is necessary while the data sharing group is running in a coexistence environment, issue the START command only for Version 8 members. Do not start the Version 7 members until the Version 8 members have completed forward log recovery. If a Version 7 member performs the group restart for a Version 8 member, Version 7 adds pages to the logical page list during the peer-forward recovery phase when it tries to apply redo log records against a release-dependent object.

Recommendation for SPUFI

When you migrate the first member of the data sharing group to Version 8, you run DSNTIJSG which rebinds SPUFI in Version 8. Binding SPUFI in Version 8 causes SPUFI to be unavailable to the Version 7 members. If you attempt to run an SQL statement on a member that has yet to migrate to Version 8, expect messages that indicate an unavailable resource.

Coexistence considerations

Coexistence considerations are similar to those you need to understand for the fallback environment. For example, objects that are frozen in fallback are generally not accessible from a down-level member (Version 7). See the fallback considerations section of Part 2 of DB2 Installation Guide for more information.

See DB2 Release Planning Guide for a discussion of all Version 8 migration considerations. Many of the migration concerns are relevant to running a data sharing group in a coexistence environment.

Version 8 migration modes: The DB2 Version 8 migration process is designed to be accomplished in phases.

- Migrating a member to DB2 Version 8 initially puts that member into Version 8 compatibility mode. In this mode, no new Version 8 function is available for use.
- To make new function available, you must put the group through the enabling new-function mode process. Before attempting this process, confirm that all started members are running DB2 Version 8. The enabling new-function mode process can only begin if all members are running Version 8.
- Assuming that the enabling new-function mode process completes successfully, the group enters Version 8 new-function mode, in which all new Version 8 function is available for use.
From Version 8 new-function mode, members cannot return to Version 8 compatibility mode, nor fall back to Version 7. The only way to recover to a previous catalog level is to perform a point-in-time recovery of both the catalog and directory.

See [DB2 Installation Guide](#) for detailed information about Version 8 new-function mode.

**Procedure to migrate the data sharing group**

*Jobs that the migration CLIST tailors:* Jobs marked with an asterisk (*) are modified only when you migrate the first member of a data sharing group.

<table>
<thead>
<tr>
<th>CLIST</th>
<th>PDS Member</th>
<th>CLIST</th>
<th>PDS Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNTIJEX*</td>
<td>DSNTIJGF</td>
<td>DSNTIJIC*</td>
<td>DSNTIJMP</td>
</tr>
<tr>
<td>DSNTIJIC*</td>
<td>DSNTIJMP</td>
<td>DSNTIJVC*</td>
<td>DSNTIJTC*</td>
</tr>
<tr>
<td>DSNTIJFT</td>
<td>DSNTIJIN*</td>
<td>DSNTIJTC*</td>
<td>DSNTIJTM*</td>
</tr>
<tr>
<td>DSNTIJFV</td>
<td>DSNTIJMV</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The DB2I CLISTs and sample jobs also get edited when the first member migrates to Version 8.

**Follow these directions carefully.** The first member of the data sharing group uses DSNTIDXA as its input member name. A subsequent member must use a previous member’s output member name as its input member name.

To migrate the data sharing group:

1. For the first member to migrate, specify the following on installation panel DSNTIPA1:
   
   INSTALL TYPE ===> MIGRATE
   DATA SHARING ===> YES
   
   DATA SET NAME(MEMBER) ===> *this member’s output member from Version 7.*
   
   INPUT MEMBER NAME ===> DSNTIDXA
   OUTPUT MEMBER NAME ===> *this member’s output PDS member*

2. On installation panel DSNTIPP2, specify 1 to indicate that this is the first member of the group to migrate.

3. Complete the installation panels. Specify parameters according to the guidelines in “The scope and uniqueness of DB2 subsystem parameters” on page 70.

4. Complete all migration steps, as described in Part 2 of [DB2 Installation Guide](#). This is the last step for the first member.

   **Important:** You must complete the migration of the first member of the data sharing group to Version 8 before starting any other members at Version 8.

5. Migrate the next member or members of the group. Specify the following on installation panel DSNTIPA1:
   
   INSTALL TYPE ===> MIGRATE
   DATA SHARING ===> YES
   
   DATA SET NAME(MEMBER) ===> *this member’s output member from Version 7.*
   
   INPUT MEMBER NAME ===> *first member’s output PDS member*
   OUTPUT MEMBER NAME ===> *this member’s output PDS member*  

6. Specify 2 on installation panel DSNTIPP2 to indicate that this is not the first member of the group to migrate.
7. **Complete the following migration steps, as described in Part 2 of \( DB2 \) Installation Guide**
   
   a. Migration Step 6: Connect DB2 to TSO.
      
      You do not have to run job DSNTIJVC for the second and subsequent members.
   
   b. Migration Step 7: Connect IMS to DB2 (optional).
   
   c. Migration Step 8: Connect CICS to DB2 (optional).
   
   
   e. Migration Step 11: Define DB2 initialization parameters.
      
      Job DSNTIJUZ only contains a subset of the steps contained in the first migrating member's DSNTIJUZ job. DSNHDECP is not reassembled for each subsequent migrating member.
   
   
   g. Migration Step 13: Define DB2 Version 8 to z/OS.
      
      Job DSNTIJMV contains a subset of the steps contained in the first migrating member's DSNTIJMV job. The steps to update the IEAAPFx and LNKLISTxx members, and the steps to update the language procedures are not included when subsequent members migrate.
   
   h. Migration Step 16: IPL z/OS.
      
      This step can be performed before migrating DB2 if you have made the appropriate updates to the z/OS libraries.
   
   

8. For each subsequent member, repeat all steps beginning with step 5 on page 99.

---

### Falling back and remigrating

When data sharing, you can fall back from Version 8 compatibility mode in one of two ways:

- Fall back one member at a time
- Fall back all members at the same time

The fallback procedure you use depends on your situation and environment. See “Mixed releases in a data sharing group” on page 92 for more information.

### Falling back

The procedure in this section describes falling back to Version 7.

Before falling back to Version 7, you must have the fallback and coexistence SPE and its prerequisite APARs installed on the Version 7 load library. If all members have already been migrated to the new release, and you are falling back one member at a time, after the first member falls back you are running in coexistence mode. Read “Mixed releases in a data sharing group” on page 92 for information about coexistence mode. For complete information about falling back to Version 7, see Part 2 of DB2 Installation Guide.

You can perform the fallback procedure on one member of the data sharing group at a time. Other members can continue to run while one member is falling back.

To fall back to Version 7:
1. If you use the automatic restart capability of z/OS and you changed the ARM policy for DB2, IRLM, or both to utilize restart light, remove the added keywords from the policy. For more information, see “Creating an automatic restart policy” on page 36.

2. If NOGROUP is used, remove it before falling back.

3. Stop DB2 on the member that is falling back or stop DB2 on all members if you are falling back all members at once.

4. Reactivate Version 7 for that member (DSNTIJFV) or for all members.

5. Reconnect TSO, IMS, CICS to Version 7.

6. Start Version 7:
   a. Enter the command START DB2:
      1) Check for indoubt units of recovery.
      2) Check for outstanding restrictive states.


8. Repeat steps 3 through 7 for each member (or all members) of the data sharing group.

9. When the last member is falling back, run job DSNTEJ0 to free plans and packages and drop objects created by the Version 8 sample programs.

**Falling back and disabling DB2 data sharing**

If you want to fall back to a Version 7 non-data sharing DB2 subsystem, preform the following procedures:

1. Disable data sharing on Version 8 as described in “Procedure to disable data sharing” on page 103. When you complete this procedure, you have a single member, which is called the surviving member.

2. Fall back to Version 7, as described in Part 2 of DB2 Installation Guide.

*Take special care when the survivor is not the originating member:* If you disable data sharing so that the surviving member is not the originating member, be sure to complete the following tasks when you fall back:

1. Tailor the DSNTIJFV job to refer to the procedures used by the surviving member.

2. Modify the ssmnMSTR procedure of the surviving member, to include the correct BDS names.

3. Modify the ssmnSPAS procedure to include the correct SUBSYS name. Modify the ssmnWLM procedures, if any, of the surviving member to include the correct DB2SSN parameter.

4. Rerun job DSNTIJUZ to update DSNZPxxx and DSNHDECP to refer to the surviving member.

**Remigrating**

You can remigrate each member or all members of a data sharing group. The method you choose depends on your situation and environment. See “Mixed releases in a data sharing group” on page 92 for more information.

If you followed the procedure in “Falling back” on page 100, simply remigrate each member of the data sharing group as described in Part 2 of DB2 Installation Guide.
Disabling and re-enabling DB2 data sharing

Disabling DB2 data sharing is a complex procedure and is very disruptive. Do not attempt to disable DB2 data sharing without a thorough understanding of the process.

Do not make disabling DB2 data sharing part of your contingency plans for handling recovery situations. For temporary bypasses to data sharing problems, try moving to one-way data sharing, which involves stopping all but one member and having that member perform the work for the group. If one-way data sharing does not work, your data sharing problem might be resolved by performing a group restart.

The disabling procedure is included in this section for completeness, but there should be very little reason to ever perform it. Situations in which it might be necessary to disable DB2 data sharing are:

• You made a strategic decision to move away from a data sharing environment.
• One-way data sharing is not working.

After you have disabled DB2 data sharing, only one member of the data sharing group can access the previously shared data. That member is called the surviving member.

Disabling DB2 data sharing

This section describes:

• “Summary of disabling procedure”
• “Procedure to disable data sharing” on page 103
• “Data recovery after disabling DB2 data sharing” on page 104.

Summary of disabling procedure

The procedure to disable data sharing ensures that the most recent versions of all pages are externalized from the group buffer pool to disk. DB2 does not use the group buffer pool after data sharing is disabled. You must ensure that data is written to disk, or else you lose data when you start DB2 after disabling data sharing.

You must also ensure that there is no need to recover data from information contained in other members' logs after you disable data sharing, as described in “Data recovery after disabling DB2 data sharing” on page 104. Other members' logs are not available to the surviving member after you disable data sharing. To prevent the surviving member from applying inconsistent updates during recovery processing, a cold start is required to disable data sharing.

If you are planning to re-enable data sharing for this group, do not change any group-wide information in the surviving member's BSDS. This includes the catalog alias name and the database password. It also includes the DDF name and password information, even if you are not going to use DDF when you re-enable DB2 data sharing. If you change any of this information, you will have to change the information in every member's BSDS before you start the group.
**Important:** Do not attempt to go through the installation process to re-enable data sharing after you have disabled it. You must use the procedure described in “Procedure to re-enable DB2 data sharing” on page 105.

**Procedure to disable data sharing**
The procedure to disable data sharing, and return to the Version 8 non-sharing environment is as follows:

1. Decide which member will be the surviving member of the group.
2. Stop all members by entering the following command for each member of the data sharing group:
   ```
   STOP DB2 MODE(QUIESCE)
   ```
3. Start the surviving member in maintenance mode by using the following command:
   ```
   START DB2 ACCESS(MAINT)
   ```
4. Make sure that data is consistent. Enter the following commands from the surviving member of the group. Do not go on to the next step until all problems are resolved.
   - **DISPLAY GROUP**
     
     Make sure that the status of the surviving member is ACTIVE and that the status of all other members is QUIESCED. If a non-surviving member’s status is not QUIESCED, take the following actions, depending on the member’s status:
     - If the member’s status is FAILED, restart and then stop the member and stop it successfully.
     - If the member has castout problems, indoubt units of recovery, or outstanding resynchronization problems, start that member in maintenance mode and fix the problem.
   - **DISPLAY UTILITY(*)**
     
     If there is any remaining utility work for any member of the group, restart that member with ACCESS(MAINT) and either stop the utility or let it finish.
   - **DISPLAY DATABASE(*) SPACENAM(*) RESTRICT**
     
     If there are any restricted table spaces or index spaces (such as write error ranges, recovery pending status, or logical page list entries), recover them from the surviving member.
5. Stop the surviving member by using the following command:
   ```
   STOP DB2 MODE(QUIESCE)
   ```
6. Stop any IRLMs that have not stopped by using the following command:
   ```
   stop irlmproc
   ```
7. Dismantle the data sharing group.
   a. Enter the following command to display the structures for the data sharing group:
      ```
      D XCF,STRUCTURE,STRNAME=grpname*
      ```
   b. For all structures that are still allocated (STATUS:ALLOCATED) and that still have connections (which appear as FAILED PERSISTENT), enter the following command to force the connections off of those structures:
      ```
      SETXCF FORCE,CONNECTION,STRNAME=strname,CONNAME=ALL
      ```
   **Important:** If your site is running z/OS with APAR OA02620 applied, you do not need to force failed-persistent connections off of the lock structure. When you forcibly deallocate the lock structure (see the next step), the system deletes failed-persistent connections to the structure for you.
c. Delete all the DB2 coupling facility structures by using the following command for each structure:

```
SETXCF FORCE,STRUCTURE,STRNAME=strname
```

d. Edit the JCL in job DSNTIJGF to point to the correct BSIDS data sets. DSNTIJGF is a change log inventory job that sets up the surviving member for a cold start.

**Important:** Do not change the hex values that appear in the change log inventory CRESTART control statement. They are not real RBA values.

e. Run job DSNTIJGF.

After you run this job, do not try to restart any of the non-surviving members. None of those members can start successfully.

8. Change the IRLM procedure to SCOPE=LOCAL.

9. Start the surviving member with ACCESS(MAINT). Specify the old DSNZP.xxx from the non-data sharing environment. If the surviving member is not the originating member, you must reassemble the surviving member's subsystem parameters, specifying the subsystem parameter DSHARE=NO in the invocation of the DSN6GRP macro. Also, comment out all steps from the DSNTIJUZ job except for those that reassemble and link-edit the subsystem parameters.

When you start DB2 after running DSNTIJGF, respond with Y to a cold start prompt (message DSN1246I on the z/OS console).

This is a cold start because DB2 increases the log RBA to a value higher than any LRSN used while sharing data. From this point on, your RBAs look like LRSNs.

10. Edit and run job DSNTIJFT, if necessary, to ensure that the surviving member's work file database is DSNDB07.

The surviving member must use DSNDB07 as its work file database. If the work file database for the surviving member is not DSNDB07, drop that work file database and run job DSNTIJFT.

11. Verify that the surviving member works by running a subset of the Version 8 installation verification sample jobs.

See the step for “Verifying Your Version 8 Subsystem” in Part 2 of DB2 Installation Guide for a list of these jobs.

12. To establish a new recovery point, take a full or incremental image copy or non-DB2 backup of all data. Run job DSNTIJIC to image copy the DB2 catalog and directory.

**Recommendation:** Perform this step as soon as possible after data sharing is disabled. See “Data recovery after disabling DB2 data sharing” for more information about recovery.

13. Stop and restart DB2 for normal unrestricted access.

**Data recovery after disabling DB2 data sharing**

After DB2 data sharing is disabled, you cannot recover to the current point or to a previous point in time if that recovery depends on any portion of the log made before you disabled data sharing. Therefore, if there are any updates to the table space between the time of the copy and the time you disabled data sharing, DB2 does not let you use that copy as the basis for recovery. This is why you should create a new recovery point as soon as possible after you have disabled data sharing.
Using the group attachment after disabling DB2 data sharing

After disabling DB2 data sharing, you can continue to use the group attachment name. There is no need to change this to the surviving member’s subsystem ID.

Re-enabling DB2 data sharing

Perform a subset of the original procedure to re-enable DB2 data sharing. You cannot use the enabling process of installation to re-enable DB2 data sharing.

The following output from the original DB2 data sharing enabling procedure remains intact after disabling DB2 data sharing and does not need to be recreated or re-specified:

- Data sharing subsystem parameters (output from the CLIST processing when enabling data sharing)
- XCF definitions
- Coupling facility definitions
- RACF definitions
- DB2 catalog and directory

Procedure to re-enable DB2 data sharing

The procedure to re-enable DB2 data sharing is:

1. Edit the JCL in job DSNTIJGF.
   
   DSNTIJGF is a change log inventory job that sets up the BSDS of the surviving member for data sharing.

2. Run job DSNTIJGF.

3. Change the IRLM procedure to SCOPE=GLOBAL.

4. Start the surviving member with the subsystem parameters that were used when data sharing was originally enabled.
   
   During startup, you are asked to start all other members that were not quiesced at the time that you disabled DB2 data sharing. You must start all these members.

5. Edit and run job DSNTIJFT on the surviving member to recreate the work file database for data sharing.
   
   See the directions in the prologue of job DSNTIJFT for information about editing this job to re-enable DB2 data sharing.

Removing members from the data sharing group

One of the features of DB2 data sharing is incremental growth, being able to add members to an existing group. However, there might be a situation in which you want to remove members from the group, either temporarily or permanently. For example, assume that your group does the job it needs to do 11 months of the year. However, a surge of additional work every December requires you to expand your capacity. You can quiesce some members of the group for those 11 months. Those members are “dormant” until you restart them.

The same principle is used to “remove” a member of the group forever. Make sure the member is quiesced cleanly, and that member can remain dormant forever. In effect, it is removed from the group.

A quiesced member (whether you intend for it to be quiesced forever or only temporarily) still appears in displays and reports. It appears in DISPLAY GROUP output with a status of QUIESCED.
What data sets to keep

When you quiesce a member, you must keep the log data sets until such time as they are no longer needed for recovery (other members might need updates that are recorded on the quiesced member's log). You must keep the BDS, too, because it contains information that points to those log data sets.

The BDS is also needed for group restart. However, if you are confident that logs for the quiesced member are no longer necessary, because that member has been quiesced for a long time or is permanently quiesced, it is possible to delete the BDS data sets. If you delete the BDS data sets, you must expect the following message during group restart:

DSNR020I -DB2A csect-name START MEMBER DB1A, OR REPLY 'NO' or QUIESCED

When you respond with QUIESCED, DB2 issues the following message:

DSNR030I -DB2A csect-name WILL CONTINUE WITHOUT THE DB1A MEMBER'S LOG,

REPLY 'YES' OR 'NO'

In summary, you must do one of the following things to ensure that you have group restart capability:

- Keep the quiesced member's BDS data sets (thus avoiding the preceding WTOR messages).
- Update your group restart procedure to ensure that operators know how to respond to the DSNR020I and DSNR030I messages.

Procedure to quiesce a member

In summary, to quiesce a member of a data sharing group:

1. Stop the member you are going to quiesce. This example assumes that you want to quiesce member DB3A:
   -DB3A STOP DB2 MODE(QUIESCE)

2. From another member, enter the following commands:
   DISPLAY GROUP
   DISPLAY UTILITY (*) MEMBER(member-name)
   DISPLAY DATABASE(*) RESTRICT
   See step 4 on page 103 for more information about using these commands. If there is no unresolved work, you can stop now. However, if you want to create an archive log, continue to the next step.

3. If there is unresolved work, or if you want to create a disaster recovery archive log (as in step 4), start the quiesced member with ACCESS(MAINT).
   -DB3A START DB2 ACCESS(MAINT)
   If there is unresolved work, resolve any remaining activity for the member, such as resolving indoubt threads, finishing or stopping utility work, and so on.

4. Optional: To create an archive log that can be sent to a disaster recovery site, archive the log for the member by entering the following command:
   -DB3A ARCHIVE LOG

5. Stop the member again with MODE(QUIESCE).
   -DB3A STOP DB2
   MODE(QUIESCE)
Chapter 4. Communicating with data sharing groups

A data sharing group can be a powerful server in your client/server environment. The group can be part of a TCP/IP network, part of an SNA network, or part of a network that uses both protocols. The group has a single-system image to requesting applications, whether requests arrive through TCP/IP or SNA. Queries can originate from any system or application that issues Structured Query Language (SQL) statements as a requester in the formats that are described by Distributed Relational Database Architecture™ (DRDA).

Recommendation: Avoid using DB2 private protocol. In addition to its support for the DRDA protocol, SNA also supports queries that originate from DB2 subsystems that use DB2 private protocol. However, using DB2 private protocol is not recommended because this protocol does not support many distributed functions, such as stored procedures. Additionally, this protocol does not support some data types, such as LOB and distinct types.

The distributed data facility (DDF) of DB2 uses TCP/IP and SNA to communicate with other DB2 subsystems. The DDF enables a DB2 subsystem to access data that is held by other database management systems. The DDF also enables the DB2 subsystem to make its own data accessible to other DB2 subsystems.

A data sharing group can support many more connections than a single member of the group can support. The DDF connections limit for a group is “n × 150 000”, where n is the number of members in the group. Thus, a group with 16 members can support 2 400 000 DDF connections.

Before reading this chapter: Be sure to read Part 3 of DB2 Installation Guide, which describes the configuration details that are specific to connecting distributed subsystems.

This chapter describes the following topics:

- “Ways to access data sharing groups”
- “TCP/IP access methods” on page 109
- “SNA access methods” on page 127
- “Preventing a member from processing requests” on page 142
- “Using the DSNJU003 utility to update the BDS” on page 142

Ways to access data sharing groups

This section provides an overview of the ways that requesters can access data sharing groups in TCP/IP and SNA networks. A requester is a client that manages connections and data access for client applications. Any product that supports the application requester protocols that are defined by DRDA can be a requester. In this chapter, the focus is on the following two requesters:

- DB2 Connect™ Enterprise Edition
  DB2 Connect Enterprise Edition is a connectivity server that concentrates and manages connections from multiple desktop clients and Web applications to DB2 database servers such as DB2 UDB for z/OS.

- DB2 UDB for z/OS
  DRDA requester support is an integral part of the DDF component of DB2.
In a connection request, a client application specifies the DB2 location name of the data sharing group to which it wants to connect. The requester then maps this location name to an actual network element that identifies the member to which to connect. In the case of TCP/IP, this network element is either a domain name or an IP address. In the case of SNA, this network element is an LU name.

Clarification: For TCP/IP, the IP address is associated with a default port. To use a different port, the requester must specify the port number in addition to the IP address. For SNA, the LU name is associated with a default transaction program name. To use a different transaction program, the requester must specify the transaction program name in addition to the LU name.

Overview of TCP/IP access methods

In TCP/IP networks, use a member dynamic VIPA or group dynamic VIPA to connect to a data sharing group. The use of dynamic VIPAs offers accessibility benefits beyond those offered by the use of member and group domain names.

A TCP/IP requester can use one of several access methods to connect to a data sharing group:

Group access
A requester uses the group’s dynamic virtual IP address (VIPA) to make an initial connection to any member of the group. Alternatively, a requester uses the group’s domain name, which maps to either a list of member IP addresses or to members’ dynamic VIPAs, to make an initial connection to any member of the group. The member that receives the connection request returns a list of members that are currently active and able to perform work. The requester uses this information to send subsequent connection requests and query requests to available members.

Member-specific access
A requester uses a location alias, which specifies one or more members of the group, to make an initial connection to one of the members that is represented by the alias. The member that receives the request returns a list of members that are currently active and able to perform work. The requester uses this information to send subsequent connection and query requests to available members that are represented by the location alias.

Single-member access
A requester uses a member’s dynamic VIPA, actual IP address, or domain name to connect and send queries to a single member of the group. All connection and query requests from that requester are directed to the same member.

Recommendation: Use a member dynamic VIPA or the group dynamic VIPA to connect to a data sharing group. The use of dynamic VIPAs offers accessibility benefits beyond those offered by the use of member and group domain names.

To achieve the highest level of application availability in a data sharing environment, configure the DB2 group and each DB2 member to use dynamic VIPA. Then route all connections through a DB2 Connect Server with Sysplex support enabled. The DB2 Connect server then manages the connections to the group using the dynamic VIPAs.

To achieve the highest level of Web application availability using JTA and JTS distributed transactions against a data sharing group, configure the DB2 group and each DB2 member to use dynamic VIPA. When the DB2 Universal JDBC Driver
Overview of SNA access methods

In SNA networks, each member of a data sharing group has its own LU name (in addition to the net ID that the member obtains from VTAM® when the DDF starts). You can also define a *generic LU name* that represents all the members in the group.

An SNA requester can use one of several access methods to connect to a data sharing group:

**Member-specific access**
A requester uses a location alias, which specifies one or more members of the group, to make an initial connection to one of the members that is represented by the alias. The member that receives the request returns a list of members that are currently active and able to perform work. The requester uses this information to send subsequent connection and query requests to available members that are represented by the location alias.

**Group-generic access**
A requester uses the group's generic LU name to make an initial connection to any member of the group. With this access method, VTAM chooses one of the group members and establishes a session with that member on behalf of the requester. Subsequent queries from the requester are also directed to that same member.

**Single-member access**
A requester uses a member’s LU name to connect and send queries to a single member of the group. All connection and query requests from that requester are directed to the same member.

Mixed TCP/IP and SNA networks

The members of a data sharing group support both TCP/IP and SNA network protocols. However, when acting as requesters, members must be configured to communicate with other data sharing groups by using either TCP/IP or SNA. The DDF uses the group's communication database (CDB) to map each remote DB2 location name to either an IP address (for TCP/IP) or an LU name (for SNA). If the same remote DB2 location name is configured for both TCP/IP and SNA, TCP/IP is used.

TCP/IP access methods

TCP/IP is a set of communication protocols that support peer-to-peer connectivity functions for both local and wide area networks. On the sending end, Transmission Control Protocol (TCP) manages the assembly of a message or file into smaller packets that are transmitted over a network. On the receiving end, TCP manages the re-assembly of those packets into the original message or file. Internet Protocol (IP) handles the routing of each packet, ensuring that packets reach the correct destination. Together, TCP and IP use the client/server model of communication to enable communication between computers and computer networks of the same or different types.

This section describes the following topics:
- “Group access” on page 110
Group access

This section provides an overview of how to use dynamic virtual IP address (VIPA) network addressing or domain name server (DNS) network addressing to access a data sharing group. Depending on the network addressing method that you plan to use, see your network administrator for the following information:

- If you plan to use dynamic VIPA network addressing:
  - Can dynamic VIPA be configured for the Parallel Sysplex?
- If you plan to use DNS network addressing:
  - Is a DNS already installed in the Parallel Sysplex?
  - What Parallel Sysplex domain name is used by the DNS?
  - Can dynamic VIPA be mapped to unique domain names in the DNS?

Dynamic VIPA network addressing

Dynamic virtual IP addressing gives you the ability to assign a specific, virtual IP address to a data sharing group and to each member of the group. This address is independent of any specific TCP/IP stack within the Parallel Sysplex. Even if a member is moved to another z/OS system, as in the case of a failure or maintenance, the member remains accessible and retains the same virtual IP address.

The Sysplex Distributor plays a role in dynamic VIPA network addressing. Because it resides on a system in the Parallel Sysplex cluster, the Sysplex Distributor can factor real-time information such as member status and Quality of Service (QoS) data with information that is obtained from Workload Manager (WLM) to ensure that the best member is chosen to serve each client request.

If, in its connection request, an application specifies a DB2 location name that maps to a group dynamic VIPA, the Sysplex Distributor dispatches the requester’s initial connection request to the active member with the most capacity, thus providing network-level workload balancing. By using the group dynamic VIPA in its connection requests, the requester ensures that workload balancing is invoked as indicated below:

- Requests are always routed to an active member, if at least one member is active.
- Requests are dynamically directed to those members with the most capacity.

**Important:** Requesters must use the group dynamic VIPA on all connection requests because a member does not bind to any IP address (INADDR_ANY) that is owned by a host. Instead, the member binds to the group and member dynamic VIPAs.

A member that receives an initial connection request returns a list of members that are currently active and able to perform work, thus providing database-level workload balancing. The list includes member IP addresses and ports, which the requester uses to connect to the member with the most capacity.
**Recommendation:** For the highest level of availability and workload balancing, use dynamic VIPA network addressing. Enable the Sysplex Distributor for network-level load balancing of initial connections, and use database-level workload balancing for load balancing of subsequent connections. Network-level load balancing ensures that every initial connection is routed to the active member with the most capacity. After member availability and capacity information is returned on the initial connection, a requester can use database-level workload balancing to choose the appropriate member for subsequent connections. For detailed information about using dynamic VIPAs with the Sysplex Distributor, see *z/OS Communications Server: IP Configuration Guide*.

**Example configuration:** [Figure 22 on page 112](#) shows the TCP/IP configuration statements that are required to set up a three-member data sharing group to route dynamic VIPA requests and perform workload balancing across the members.

In this figure:
- Vx represents the group dynamic VIPA, and V1, V2, and V3 represent the members’ dynamic VIPAs.
- The group dynamic VIPA and the member-specific dynamic VIPAs are defined on each TCP/IP stack.
- The SHAREPORT option is required; without this option, TCP/IP would not allow multiple listeners on the DRDA port.
- The BIND (SPECIFIC) option directs DB2 to implicitly use dynamic VIPA network addressing.
- VIPADYNAMIC statements:
  - The group dynamic VIPA must be defined with the VIPADEFINE and VIPADISTRIBUTE statements on the TCP/IP stack that is associated with the system (DB2A) on which the Sysplex Distributor executes.
  - The group dynamic VIPA must be defined with the VIPABACKUP statement on the TCP/IP stacks that are associated with the systems (DB2B and DB2C) on which the Sysplex Distributor does not execute.
  - To allow for failover, the member-specific dynamic VIPAs are defined with the VIPARANGE statement on all TCP/IP stacks.

See *z/OS Communications Server: IP Configuration Guide* for more information about VIPADYNAMIC statements.
Step 1. The initial connection uses the group dynamic VIPA (Vx). Port 446 is identified as the DRDA port in each member’s PORT statements.

Step 2. The Sysplex Distributor dispatches the initial connection request to the member with the lightest workload (DB2B).

Step 3. Resynchronization information and a list of members in the data sharing group are returned to the requester.

Step 4. Database-level workload balancing connections are established.

Configuration Requirements: Dynamic VIPA network addressing requires you to define multiple dynamic VIPAs, one for the data sharing group and one for each member of the group.

- Group dynamic VIPA
  
The group dynamic VIPA is common to all members of the group, and is used to make the initial connection to a member of the group.
Define the group dynamic VIPA at the end of each DRDA port number entry in
the PORT statement of each member’s TCP/IP profile data set
(PROFILE.TCPIP). For example:

    PORT
    446 TCP DB2ADIST SHAREPORT BIND Vx
    446 TCP DB2BDIST SHAREPORT BIND Vx
    446 TCP DB2CDIST SHAREPORT BIND Vx

You can configure the Sysplex Distributor to use the group dynamic VIPA to
provide network-level workload balancing among members. See z/OS
Communications Server: IP Configuration Guide for detailed information about
configuring the Sysplex Distributor.

- Member-specific dynamic VIPAs
  Every member of the group needs a unique dynamic VIPA, which is used to
directly connect to a particular member after the initial connection.
Configure dynamic VIPAs for the primary member and for any backup members
that can be started. To define each member’s dynamic VIPA, specify a RESYNC
PORT statement in the member’s TCP/IP profile data set (PROFILE.TCPIP). For
example:

    PORT
    5001 TCP DB2ADIST BIND V1
    5002 TCP DB2BDIST BIND V2
    5003 TCP DB2CDIST BIND V3

Also ensure that the following conditions exist before making a data sharing group
available for group access that uses dynamic VIPA network addressing:
- All members of the group are configured with dynamic VIPAs before starting
  the DDF on any member of the group.
- All members of the group are configured with dynamic VIPAs before remote
  connections are made to any member of the group.
- All members are running OS/390 Version 2, Release 10 or later.

Before moving to dynamic VIPA network addressing from another access method,
have all members check for the existence of indoubt threads. If any indoubt
threads exist, resolve them before moving. To check for indoubt threads, use the
DISPLAY THREAD command. See DB2 Command Reference for detailed information
about this command.

Preparing for failure recovery: Using dynamic VIPA is the most flexible way to
be prepared for DB2 or system failure. If at least one member of a data sharing
group is currently active and can perform work, requesters’ connection attempts
do not fail.

If a member suffers a failure or the underlying z/OS system suffers an outage,
automation software such as z/OS Automatic Restart Manager (ARM) can restart
the member on a different z/OS system. When this happens, the member-specific
dynamic VIPA is also moved to the new system and automatic recovery of any
indoubt threads is performed, thereby enabling requesters to access the member on
the new z/OS system.

DNS network addressing
A domain name server (DNS) provides standard name resolution services for IP
applications in a Parallel Sysplex cluster, converting domain names into IP
addresses.

Chapter 4. Communicating with data sharing groups  113
If, in its connection request, an application specifies a DB2 location name that maps to a group domain name, the requester queries the DNS with a call to `gethostbyname`. Either the DNS resolves the name to a member’s IP address, or it resolves the name to a list of member IP addresses. The DNS returns the IP address or list of IP addresses to the requester, which then uses it to make an initial connection to a member of the data sharing group.

The member that receives the connection request works in conjunction with Workload Manager (WLM) to return to the requester a list of members that are currently active and can perform work. The list includes member IP addresses and ports, and a weight for each active member that indicates the member’s current capacity. The requester uses the returned information to connect to the member with the most capacity.

**Recommendation:** If your DNS supports it, configure the DNS to return a list of member IP addresses instead of returning a single IP address. Because a DNS does not know whether a member is active when it returns the member’s IP address to the requester, the DNS might return the IP address of an inactive member. Using this IP address on a subsequent connection request causes the request to fail. In contrast, if you configure the DNS to return a list of member IP addresses, the connection request can be retried using another IP address on the list. When the DSN retries the connection request, it will run through the list once, searching for the active member.

**Example configuration:** The configuration in Figure 23 is an example of group access that uses DNS network addressing. Applications use the group’s DB2 location name (DB2A) to direct connection requests to the group.

![Diagram of TCP/IP network configuration](image)

**Figure 23. Example TCP/IP network configuration (DNS network addressing)**

**Note:** When a DB2 for z/OS system is a member of a data sharing group, DB2 cannot be utilized as a DRDA XA server. As a result, the XA Transaction Manager cannot directly access the DB2 for z/OS DRDA server. Instead, you must configure
Member-specific access

A DB2 location name represents all the members of a data sharing group. In contrast, member-specific access uses location aliases that map to the domain names or IP addresses of one or more group members. Requesters can use location aliases to bypass default TCP/IP workload balancing and to establish connections with one or more members. Workload is balanced among members at the requester’s discretion.

With member-specific access, a requester uses a location alias to make an initial connection to one of the members that is represented by the alias. The member that receives the connection request works in conjunction with Workload Manager (WLM) to return a list of members that are currently active and able to perform work. The list includes member IP addresses and ports, and a weight for each active member that indicates the member’s current capacity. (Be aware that the list can include members that are not represented by the location alias.) The requester uses this information to connect to the member or members with the most capacity that are also associated with the location alias.

Single-member access

Single-member access is the same method that is used to access DB2 in non-data sharing environments. For detailed information about connecting to DB2 by using this method, consult [DB2 Installation Guide](#). With single-member access, a requester uses the real or virtual IP address in its TCP/IP configuration to connect to a specific member of the group. (Alternatively, a requester’s TCP/IP configuration can contain a domain name, which resolves to an IP address.) The requester sends all connection and query requests to this same member.

**DB2 Connect:** Configure DB2 Connect to use single-member access by disabling its Sysplex support. If you do not disable Sysplex support, DB2 Connect uses DRDA workload balancing by default to allocate requests among the members of the group.

**DB2 UDB for z/OS:** Configure DB2 UDB for z/OS to use single-member access by creating and using a location alias that represents only one member of the data sharing group. Otherwise, DB2 UDB for z/OS uses DRDA workload balancing by default to allocate requests among the members of the group.

**Recommendation:** Do not use single-member access for the following reasons:

- It is dependent on the specified member being operational.
- If the specified member is restarted on another CPC, its IP address changes, thereby invalidating the IP address in requesters’ TCP/IP configurations.
- It provides no workload balancing across the members of a data sharing group.

Setting up DB2 UDB for z/OS as a requester

When DB2 UDB for z/OS acts as a requester, it can connect applications that run on the z/OS system to remote database servers, including DB2 UDB for z/OS, DB2 UDB for OS/390, DB2 UDB for iSeries, and DB2 Server for VSE & VM.
In its role as a requester, DB2 UDB for z/OS accepts DB2 location names and translates them into TCP/IP addresses. It uses the communications database (CDB) to register location names and their corresponding network parameters. The data that is stored in the CDB enables DB2 to pass the required information when making distributed database requests over TCP/IP connections.

Much of the processing in a distributed database environment requires the exchange of messages with other locations in the network. For this processing to be performed correctly, you must:

1. Define the DB2 UDB for z/OS requester to the local TCP/IP system.
   
   See DB2 Installation Guide for detailed instructions on defining the DB2 requester to the local TCP/IP system.

2. Identify the remote data sharing groups to which applications can connect.

**Identifying remote data sharing groups**

When an application requests data from a remote data sharing group, the DB2 UDB for z/OS requester searches the CDB for information about the remote group. Recall that DB2 uses the CDB to store information about how to communicate with remote groups, and that the DDF uses the CDB to map DB2 location names and location aliases to IP addresses.

Group access requires:

- A DB2 location name entry in the SYSIBM.LOCATIONS table
- A conversational security requirements entry in the SYSIBM.IPNAMES table

Member-specific access requires:

- Location alias entries in the SYSIBM.LOCATIONS table
- Conversational security requirements entries in the SYSIBM.IPNAMES table
- Location alias member entries in the SYSIBM.IPLIST table

Single-member access requires:

- A location alias entry in the SYSIBM.LOCATIONS table
- A conversational security requirements entry in the SYSIBM.IPNAMES table
- A location alias member entry in the SYSIBM.IPLIST table

The following sections provide more detail about each of these tables.

**SYSIBM.LOCATIONS**: SYSIBM.LOCATIONS maps the location names in connection requests to the port numbers (or service names) of remote systems. SYSIBM.LOCATIONS must contain at least one row for each remote group, depending on the access method that is used.

- For group access, the LOCATION column of the row contains the group’s DB2 location name.
- For member-specific access, the LOCATION column of each row contains a location alias that identifies one, several, or all members of the group.
- For single-member access, the LOCATION column of the row contains a location alias that identifies one member of the group.

**Tip**: You can specify a case-sensitive service name, instead of a port number, in the PORT column of the SYSIBM.LOCATIONS table.

**SYSIBM.IPNAMES**: SYSIBM.IPNAMES maps location names to the IP addresses or domain names of remote systems. It also maps location names to the network.
security information required by the remote system. SYSIBM.IPNames must
contain at least one row for each remote group, depending on the access method
that is used.

- For group access, the IPADDR column of the row contains the group’s domain
  name.
- For member-specific access, the LINKNAME column of each row contains a link
  name, and the IPADDR column of each row is blank.
- For single-member access, the LINKNAME column of the row contains a link
  name, and the IPADDR column of the row is blank.

**SYSIBM.IPLIST:** SYSIBM.IPLIST supports member-specific access to a remote
data sharing group by enabling you to associate location aliases with one or more
members of the group. SYSIBM.IPLIST must contain a row for every member that
is associated with a location alias. For members that are associated with multiple
location aliases, insert multiple rows.

- For member-specific access, the LINKNAME column of each row contains a
  location alias and the IPADDR column of that row contains the dynamic VIPA or
domain name of a member that is associated with that location alias.
- For single-member access, the LINKNAME column of the row contains a
  location alias and the IPADDR column of that row contains the dynamic VIPA or
domain name of the member that is associated with that location alias.

**Example:** Assume that a remote data sharing group has six members. With
member-specific access, you might have two location aliases, one of which is
associated with three of the members and the other of which is associated with
two of the members. In this example, you insert five rows into the SYSIBM.IPLIST
table, three rows for the members that are associated with the first location alias
and two rows for the members that are associated with the second location alias.

**Sending requests:** When sending a request, DB2 uses the value in the
LINKNAME column of the SYSIBM.LOCATIONS table to determine which
network protocol to use.

- If DB2 finds the same value in the LINKNAME column of the
  SYSIBM.IPNames table, it uses TCP/IP.
- If DB2 finds the same value in the LUNAME column of the SYSIBM.LUNAMES
  table, it uses SNA.
- If DB2 finds the same value in both SYSIBM.IPNames and SYSIBM.LUNAMES,
  it uses TCP/IP.

**Updating the communications database:** You can update the CDB while the DDF
is active. Changes to SYSIBM.LOCATIONS, SYSIBM.IPNames, and
SYSIBM.IPLIST take effect in the following manner:

- If the DDF has not yet tried to communicate with a particular remote group,
  updates take effect when the DDF attempts to communicate with that group.
- If the DDF has already attempted to communicate with a particular remote
  group, updates take effect the next time the DDF is started.

Updates do not affect communication already in progress; existing communication
continues as if the updates had not occurred.

**Configuring data sharing groups as TCP/IP servers**

Configuring a DB2 UDB for z/OS data sharing group as a TCP/IP server is very
simple. The process consists of these steps:
1. Specifying the DRDA port number on which all members listen for incoming SQL requests
2. Specifying a unique resynchronization port number for each member
3. Registering the group’s domain name and each member’s domain name in the domain name server (DNS)
4. Designating subsets of members, if you want to limit the members to which DRDA requesters can connect
5. Specifying a generic LU name for the data sharing group, if you use RACF PassTickets

**Specifying the DRDA port number**

DB2 requires that all members of a data sharing group use the same, well-known port number to receive incoming SQL requests. 446 is the recommended DRDA port number. DRDA port numbers are stored in the bootstrap data sets (BSDSs) of members.

Use installation panel DSNTIP5 or the DSNJU003 (change log inventory) utility to specify the DRDA port number (PORT) for each member. Specify the same DRDA port number for all members of the same data sharing group. See Part 2 of [DB2 Utility Guide and Reference](http://www.ibm.com) for complete information about the DSNJU003 utility.

The network administrator must also register the DRDA port number with TCP/IP on each member’s z/OS system.

**Reserving the DRDA port:** If a member of a data sharing group is restarted on another CPC, the TCP/IP on that system must be configured to allow the member to use the DRDA port. To ensure this, reserve the DRDA port on each system for the DB2 DDF address space by assigning the port to every member that could conceivably start on that system. By explicitly assigning this port, you prevent other programs from using the DRDA port number.

On each system, replicate the TCP/IP PORT configuration profile statement shown here:

```
PORT
  446 TCP DB1ADIST SHAREPORT
  446 TCP DB2ADIST SHAREPORT
  446 TCP DB3ADIST SHAREPORT
  446 TCP DB4ADIST SHAREPORT
```

**Recommendation:** Specify the SHAREPORT option, as shown in the TCP/IP PORT configuration profile statement. Specifying this option configures TCP/IP to allow multiple listeners on the DRDA port (port 446). As client connection requests arrive for this port, TCP/IP distributes them across the members. TCP/IP selects the member with the fewest number of connections (both active and queued) at the time that the request is received.

The member chosen by TCP/IP receives all of the DRDA server’s workload for that TCP/IP instance, leaving the other members with no TCP/IP server threads for DRDA. This is transparent to the DRDA clients if the member that is processing the TCP/IP requests does not reach the MAX REMOTE CONNECTED thread limit. If this limit is reached, the client’s connection request is rejected.

**Tip:** After you resolve a failure situation, move the member back to its original CPC.
Specifying the resynchronization port numbers

DB2 requires that each member of a data sharing group have a resynchronization port number that is unique within the Parallel Sysplex. In the event of a failure, this unique port number allows a requester to reconnect to the correct member so that units of work that require two-phase commit can be resolved.

Use the DSNJU003 (change log inventory) utility to specify a unique resynchronization port number (RESYNC) for each member of the group. See Part 2 of DB2 Utility Guide and Reference for complete information about the DSNJU003 utility.

The network administrator must also register the resynchronization port number with TCP/IP on each member's z/OS system.

Registering names in the domain name server

You should always register the domain name of a data sharing group and the domain names of its members with the domain name server (DNS). Doing so allows requesters to access members by using group and member domain names. Registering the domain names is also recommended for resynchronization for two-phase commit processing, in case a member comes down and is subsequently restarted on another z/OS system.

Register the following names:

- Domain name of the data sharing group
  You can verify that you have the correct name to register in the DNS by starting the DDF. The name that you must register appears in the domain name field of the DSNL004I message.
- Domain name of each group member
  DB2 uses member domain names to resolve indoubt threads.

See z/OS Communications Server: IP Configuration Guide for more information about registering names in the DNS.

Designating subsets of members

DB2 UDB for z/OS allows you to define subsets of data sharing group members in TCP/IP networks. By designating subsets of members, you can:

- Limit the members to which DRDA requesters can connect.
  System and database administrators might find this useful for any number of purposes.
- Ensure that initial connections are established only with members that belong to the specified subset.
  Without subsets, requesters can make initial and subsequent connections to any member of the data sharing group.
- Provide requesters with information about only those members in the subset.
  With subsets, a member that receives an initial connection request can return to the requester a list of members that are currently active, able to perform work, and represented by the location alias.

To specify that a member belongs to a subset, use the alias-port parameter of the ALIAS option of the DSNJU003 (change log inventory) utility. By following the location alias with a TCP/IP port number that is a decimal number between 1 and 65534, you indicate that the DB2 subsystem is a member of the specified data sharing group subset. When DDF starts, it performs the following tasks:
• DDF registers any subset location aliases with z/OS Workload Manager (WLM). The list of members in the subset is managed automatically by z/OS.
• DDF adds the TCP/IP port numbers of the subset location aliases to a TCP/IP SELECT socket call for the SQL request listener. Doing so enables the Sysplex Distributor to send requests that are intended for subset members to only those members that belong to the subset. It also enables members of the subset to respond to those requests.

Specifying a generic LU name for the data sharing group
If you use RACF PassTickets for security, you must define a generic LU name for the data sharing group. Use the DB2 GENERIC LUNAME parameter on installation panel DSNTIPR or use the DSNJU003 (change log inventory) utility to define the generic LU name. The generic LU name is used to generate a valid PassTicket for the data sharing group. See Part 3 (Volume 1) of DB2 Administration Guide for more information about using RACF PassTickets.

Connecting distributed partners in a TCP/IP network
This section shows how to configure DB2 Connect requesters and DB2 UDB for z/OS requesters to access remote data sharing groups in TCP/IP networks. For information about how to configure other types of requesters, consult the product documentation for your specific requester. This section includes:
• “Configuring a DB2 Connect requester to use group access” on page 121
• “Configuring a DB2 requester to use group access” on page 122
• “Configuring a DB2 requester to use member-specific access” on page 123

Configuring a DB2 Connect requester to use group access
This section outlines the most important steps in configuring a DB2 Connect requester to use group access. For detailed information about these steps, and for complete information about DB2 Connect, refer to IBM DB2 Connect User’s Guide.
1. Use the Configuration Assistant to update the database directories that DB2 Connect uses to manage database connection information.
   Important: You must enable Sysplex support for the data sharing group by specifying the SYSPLEX parameter in the parameter string of the target database name in the DCS directory.
2. If you want applications to be able to update data in multiple remote database servers with guaranteed integrity, use the DB2 Control Center to enable and test multisite updates.
3. Bind the DB2 Connect utilities, and any applications that are developed using embedded SQL, to the databases with which they operate.
   See IBM DB2 Connect User’s Guide for detailed bind instructions.

Configuring a DB2 requester to use group access
This section describes how to update the CDB of a DB2 UDB for z/OS requester to use group access. Recall that dynamic VIPA network addressing uses dynamic virtual IP addresses (VIPAs) and DNS network addressing uses domain names (which can resolve to dynamic VIPAs).

To enable group access to a remote data sharing group:
1. Identify the DB2 location name of the remote group.
Insert the group’s DB2 location name into the LOCATION column of the requestor’s SYSIBM.LOCATIONS table.

2. Identify the dynamic VIPA or domain name of the remote group, and specify the security definitions for conversations with group members.
   Insert the group’s dynamic VIPA or domain name into the IPADDR column of the requestor’s SYSIBM.IPNAMES table.

3. Map the DB2 location name of the remote group to its dynamic VIPA or domain name.
   Insert the same link name into the LINKNAME columns of the requestor’s SYSIBM.LOCATIONS and SYSIBM.IPNAMES tables. Be sure to update only those rows that are associated with the remote group.

Example of group access that uses dynamic VIPA network addressing: This example provides sample SQL statements for enabling group access that uses dynamic VIPA network addressing. It also shows the results of those statements in the form of table excerpts. This example assumes that a remote data sharing group exists with a group location name of DB2A and a group dynamic VIPA of Vx.

SQL statements:

--- General-use Programming Interface ---

The following SQL statements update the CDB of a DB2 UDB for z/OS requester to use group access (dynamic VIPA network addressing):

1. This statement identifies the DB2 location name of the remote group:
   ```sql
   INSERT INTO SYSIBM.LOCATIONS (LOCATION, PORT)
   VALUES ('DB2A', '446');
   ```

2. This statement identifies the dynamic VIPA of the remote group and specifies security definitions for conversations with the group’s members:
   ```sql
   INSERT INTO SYSIBM.IPNAMES (SECURITY_OUT, USERNAMES, IPADDR)
   VALUES ('A', ' ', 'Vx');
   ```

3. These statements map the DB2 location name of the remote group to the dynamic VIPA of the remote group:
   ```sql
   UPDATE SYSIBM.LOCATIONS
   SET LINKNAME='DB2ALINK'
   WHERE LOCATION='DB2A';
   
   UPDATE SYSIBM.IPNAMES
   SET LINKNAME='DB2ALINK'
   WHERE IPADDR='Vx';
   ```

--- End of General-use Programming Interface ---

Table excerpts:

An excerpt of the SYSIBM.LOCATIONS table would look like Table 20.

**Table 20.** DB2 location name of a remote data sharing group in a DB2 requester’s SYSIBM.LOCATIONS table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>LINKNAME</th>
<th>PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2A</td>
<td>DB2ALINK</td>
<td>446</td>
</tr>
</tbody>
</table>
An excerpt of the SYSIBM.IPNames table would look like Table 21.

Table 21. Dynamic VIPA of a remote data sharing group in a DB2 requester's SYSIBM.IPNames table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LINKNAME</th>
<th>SECURITY_OUT</th>
<th>USERNAMES</th>
<th>IPADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2ALINK</td>
<td>A</td>
<td>Vx</td>
<td></td>
</tr>
</tbody>
</table>

The following SQL statement connects to the remote group:

CONNECT TO DB2A;

Example of group access that uses DNS network addressing: This example provides sample SQL statements for enabling group access that uses DNS network addressing. It also shows the results of those statements in the form of table excerpts. This example assumes that a remote data sharing group exists with a DB2 location name of DB2A and a group domain name of DB2A.SYSPLEX1.IBM.COM. (Note that this example is identical to the previous one, if you assume that the domain name in this example maps to a dynamic VIPA.)

SQL statements:

Use the following SQL statements to update the CDB of a DB2 UDB for z/OS requester to use group access (DNS network addressing):

1. This statement identifies the DB2 location name of the remote group:
   
   ```sql
   INSERT INTO SYSIBM.LOCATIONS (LOCATION, PORT)
   VALUES ('DB2A', '446');
   
   UPDATE SYSIBM.LOCATIONS
   SET LINKNAME='DB2ALINK'
   WHERE LOCATION='DB2A';
   
   UPDATE SYSIBM.IPNames
   SET LINKNAME='DB2ALINK'
   WHERE IPADDR='DB2A.SYSPLEX1.IBM.COM';
   ```

Table excerpts:

An excerpt of the SYSIBM.LOCATIONS table would look like Table 22.

Table 22. DB2 location name of a remote data sharing group in a DB2 requester's SYSIBM.LOCATIONS table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>LINKNAME</th>
<th>PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2A</td>
<td>DB2ALINK</td>
<td>446</td>
</tr>
</tbody>
</table>
An excerpt of the `SYSIBM.IPNAMES` table would look like:

Table 23. Domain name of a remote data sharing group in a DB2 requester's SYSIBM.IPNAMES table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LINKNAME</th>
<th>SECURITY_OUT</th>
<th>USERNAMES</th>
<th>IPADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2ALINK</td>
<td>A</td>
<td></td>
<td>DB2A.SYSPLEX1.IBM.COM</td>
</tr>
</tbody>
</table>

The following SQL statement connects to the remote group:

```
CONNECT TO DB2A;
```

**Configuring a DB2 requester to use member-specific access**

This section describes how to update the CDB of a DB2 UDB for z/OS requester to use member-specific access.

To enable member-specific access to a remote data sharing group:

1. Define one or more location aliases that identify different sets of members.
   Insert each location alias into the LOCATION column of the requester’s `SYSIBM.LOCATIONS` table.

2. Map each location alias to the security definitions for conversations with each set of members.
   Insert the same link names into the LINKNAME columns of the requester’s `SYSIBM.LOCATIONS` and `SYSIBM.IPNAMES` tables. In the `SYSIBM.IPNAMES` table, insert the appropriate security definitions for your site, and leave the `IPADDR` column blank.

3. Map each location alias to the domain names or dynamic VIPAs of the data sharing members in that set.
   For each member of a set, insert the domain name or dynamic VIPA of the member into the `IPADDR` column of the requester’s `SYSIBM.IPLIST` table.

4. Define location aliases on the server (remote data sharing group).
   For each location alias on the requester’s side, you must define a location alias at the server by using the DSNJU003 (change log inventory) utility. See “Using the DSNJU003 utility to update the BDS9” on page 142 for more information about the DSNJU003 utility.

**Example of member-specific access that uses dynamic VIPA network addressing:**

This example provides sample SQL statements for enabling member-specific access that uses dynamic VIPAs. It also shows the results of those statements in the form of table excerpts. This example assumes that a remote data sharing group exists with a DB2 location name of DB2A and three members with dynamic VIPAs V1, V2, and V3. Location aliases that are to be defined are DB2B (with members V1 and V2), DB2C (with members V2 and V3), and DB2D (with member V1).

**SQL statements:**

1. These statements define location aliases that identify different sets of members:

   ```sql
   INSERT INTO SYSIBM.LOCATIONS (LOCATION, PORT) VALUES ('DB2B', '446');
   ```
INSERT INTO SYSIBM.LOCATIONS (LOCATION, PORT)
VALUES ('DB2C', '446');

INSERT INTO SYSIBM.LOCATIONS (LOCATION, PORT)
VALUES ('DB2D', '446');

2. These statements map each location alias to the security definitions for
   conversations with each set of members:

UPDATE SYSIBM.LOCATIONS
SET LINKNAME='ALIASB'
WHERE LOCATION='DB2B';

INSERT INTO SYSIBM.IPNAMES (LINKNAME)
VALUES ('ALIASB')

UPDATE SYSIBM.LOCATIONS
SET LINKNAME='ALIASC'
WHERE LOCATION='DB2C';

INSERT INTO SYSIBM.IPNAMES (LINKNAME)
VALUES ('ALIASC')

UPDATE SYSIBM.LOCATIONS
SET LINKNAME='ALIASD'
WHERE LOCATION='DB2D';

INSERT INTO SYSIBM.IPNAMES (LINKNAME)
VALUES ('ALIASD')

3. These statements map each location alias to the dynamic VIPAs of the data
   sharing members in that set:

INSERT INTO SYSIBM.IPLIST (LINKNAME, IPADDR)
VALUES ('ALIASB', 'V1')

INSERT INTO SYSIBM.IPLIST (LINKNAME, IPADDR)
VALUES ('ALIASB', 'V2')

INSERT INTO SYSIBM.IPLIST (LINKNAME, IPADDR)
VALUES ('ALIASC', 'V2')

INSERT INTO SYSIBM.IPLIST (LINKNAME, IPADDR)
VALUES ('ALIASC', 'V3')

INSERT INTO SYSIBM.IPLIST (LINKNAME, IPADDR)
VALUES ('ALIASD', 'V1')

End of General-use Programming Interface

Table excerpts:

An excerpt of the SYSIBM.LOCATIONS table would look like Table 24.

Table 24. Location aliases for a remote data sharing group in a DB2 requestor's
SYSIBM.LOCATIONS table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>LINKNAME</th>
<th>PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2B</td>
<td>ALIASB</td>
<td>446</td>
</tr>
<tr>
<td>DB2C</td>
<td>ALIASC</td>
<td>446</td>
</tr>
<tr>
<td>DB2D</td>
<td>ALIASD</td>
<td>446</td>
</tr>
</tbody>
</table>

Location aliases are not associated with the dynamic VIPA of a remote data sharing
group in a DB2 requestor’s SYSIBM.IPNAMES table. An excerpt of the
SYSIBM.IPNAMES table would look like Table 25 on page 125.
Table 25. Excerpt of the SYSIBM.IPNAMES table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LINKNAME</th>
<th>IPADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASB</td>
<td></td>
</tr>
<tr>
<td>ALIASC</td>
<td></td>
</tr>
<tr>
<td>ALIASD</td>
<td></td>
</tr>
</tbody>
</table>

An excerpt of the SYSIBM.IPLIST table would look like Table 26.

Table 26. Dynamic VIPAs of the members that are associated with each location alias in a DB2 requester's SYSIBM.IPLIST table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LINKNAME</th>
<th>IPADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASB</td>
<td>V1</td>
</tr>
<tr>
<td>ALIASB</td>
<td>V2</td>
</tr>
<tr>
<td>ALIASC</td>
<td>V2</td>
</tr>
<tr>
<td>ALIASC</td>
<td>V3</td>
</tr>
<tr>
<td>ALIASD</td>
<td>V1</td>
</tr>
</tbody>
</table>

The following SQL statement connects to the remote group using a location alias:

CONNECT TO DB2C;

Example of member-specific access that uses DNS network addressing: This example provides sample SQL statements for enabling member-specific access that uses a domain name server (DNS). It also shows the results of those statements in the form of table excerpts. This example assumes the existence of a remote data sharing group with a DB2 location name of DB2A and three members with names LUDB2AR, LUDB2B, and LUDB2C. Location aliases that are to be defined are DB2B (with members LUDB2AR and LUDB2B), DB2C (with members LUDB2B and LUDB2C), and DB2D (with member LUDB2AR).

SQL statements:

<table>
<thead>
<tr>
<th>General-use Programming Interface</th>
</tr>
</thead>
</table>

Use the following SQL statements to update the CDB of a DB2 UDB for z/OS requester to use member-specific access (DNS network addressing):

1. These statements define location aliases that identify different sets of members:

   INSERT INTO SYSIBM.LOCATIONS (LOCATION, PORT) VALUES ('DB2B', '446');
   INSERT INTO SYSIBM.LOCATIONS (LOCATION, PORT) VALUES ('DB2C', '446');
   INSERT INTO SYSIBM.LOCATIONS (LOCATION, PORT) VALUES ('DB2D', '446');

2. These statements map each location alias to the security definitions for conversations with each set of members:

   UPDATE SYSIBM.LOCATIONS
   SET LINKNAME='ALIASB'
   WHERE LOCATION='DB2B';
   INSERT INTO SYSIBM.IPNAMES (LINKNAME, ) VALUES ('ALIASB')
UPDATE SYSIBM.LOCATIONS
    SET LINKNAME='ALIASC'
    WHERE LOCATION='DB2C';

INSERT INTO SYSIBM.IPNAMES (LINKNAME) VALUES ('ALIASC')

UPDATE SYSIBM.LOCATIONS
    SET LINKNAME='ALIASD'
    WHERE LOCATION='DB2D';

INSERT INTO SYSIBM.IPNAMES (LINKNAME) VALUES ('ALIASD')

3. These statements map each location alias to the domain names of the data sharing members in that set:

INSERT INTO SYSIBM.IPLIST (LINKNAME, IPADDR)
    VALUES ('ALIASB', 'LUDB2AR.DB2PLEX.SYSPLEX.COM')

INSERT INTO SYSIBM.IPLIST (LINKNAME, IPADDR)
    VALUES ('ALIASC', 'LUDB2C.DB2PLEX.SYSPLEX.COM')

INSERT INTO SYSIBM.IPLIST (LINKNAME, IPADDR)
    VALUES ('ALIASD', 'LUDB2AR.DB2PLEX.SYSPLEX.COM')

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>LINKNAME</th>
<th>PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2B</td>
<td>ALIASB</td>
<td>446</td>
</tr>
<tr>
<td>DB2C</td>
<td>ALIASC</td>
<td>446</td>
</tr>
<tr>
<td>DB2D</td>
<td>ALIASD</td>
<td>446</td>
</tr>
</tbody>
</table>

End of General-use Programming Interface

Table excerpts:

An excerpt of the SYSIBM.LOCATIONS table would look like Table 27.

Table 27. Location aliases for a remote data sharing group in a DB2 requester's SYSIBM.LOCATIONS table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>LINKNAME</th>
<th>PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2B</td>
<td>ALIASB</td>
<td>446</td>
</tr>
<tr>
<td>DB2C</td>
<td>ALIASC</td>
<td>446</td>
</tr>
<tr>
<td>DB2D</td>
<td>ALIASD</td>
<td>446</td>
</tr>
</tbody>
</table>

Location aliases are not associated with the dynamic VIPA of a remote data sharing group in a DB2 requester's SYSIBM.IPNAMES table. An excerpt of the SYSIBM.IPNAMES table would look like Table 28.

Table 28. Excerpt of the SYSIBM.IPNAMES table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LINKNAME</th>
<th>IPADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASB</td>
<td></td>
</tr>
<tr>
<td>ALIASC</td>
<td></td>
</tr>
<tr>
<td>ALIASD</td>
<td></td>
</tr>
</tbody>
</table>

An excerpt of the SYSIBM.IPLIST table would look like Table 29.

Table 29. Domain names of the members that are associated with each location alias in a DB2 requester's SYSIBM.IPLIST table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LINKNAME</th>
<th>IPADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASB</td>
<td>LUDB2AR.DB2PLEX.SYSPLEX.COM</td>
</tr>
</tbody>
</table>
Table 29. Domain names of the members that are associated with each location alias in a DB2 requester's SYSIBM.IPLIST table (continued). Not all columns are shown.

<table>
<thead>
<tr>
<th>LINKNAME</th>
<th>IPADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASB</td>
<td>LUDB2B.DB2PLEX.SYSPLEX.COM</td>
</tr>
<tr>
<td>ALIASC</td>
<td>LUDB2B.DB2PLEX.SYSPLEX.COM</td>
</tr>
<tr>
<td>ALIASC</td>
<td>LUDB2C.DB2PLEX.SYSPLEX.COM</td>
</tr>
<tr>
<td>ALIASD</td>
<td>LUDB2AR.DB2PLEX.SYSPLEX.COM</td>
</tr>
</tbody>
</table>

The following SQL statement connects to the remote group using a location alias:

```
CONNECT TO DB2C;
```

### SNA access methods

*Systems Network Architecture (SNA)* is a proprietary IBM architecture that describes the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks. SNA contains several functional layers and includes the following components:

- An application programming interface (API) called Virtual Telecommunications Access Method (VTAM)
- A communications protocol for the exchange of control information and data
- A data link layer called Synchronous Data Link Control (SDLC)

SNA also includes the concept of nodes, which can contain both physical units that provide certain setup functions, and logical units (LUs), each of which is associated with a particular network transaction.

A data sharing group can simultaneously serve requesters that use member-specific access, group-generic access, and single-member access. As a server, the group is flexible about the access methods that requesters use. A requester can use member-specific access in one session and use group-generic access in another session. However, as a requester, the group always uses the same access method when communicating with a particular server.

**Recommendation:** Configure a data sharing group to use member-specific access whenever possible, unless your requester does not support it. It provides better workload balancing and two-phase commit support than group-generic access.

This section discusses the following topics:

- "Member-specific access"
- "Group-generic access" on page 128
- "Single-member access" on page 129

It also includes procedures for:

- "Configuring data sharing groups as servers" on page 132
- "Configuring data sharing groups for group-generic access" on page 132
- "Setting up DB2 UDB for z/OS as a requester" on page 129
- "Connecting distributed partners in an SNA network" on page 137

### Member-specific access

A DB2 location name represents all the members of a data sharing group. In contrast, member-specific access uses location aliases that map to the LU names of
one or more group members. Requesters can use location aliases to bypass VTAM workload balancing by establishing sessions with one or more members. Workload is balanced among members at the requester's discretion.

With member-specific access, a requester uses a location alias to make an initial connection to one of the members that is represented by the alias. The DB2 Sysplex transaction program (DB2 STP) works in conjunction with the z/OS Workload Manager (WLM) to return a list of members that are currently active and can perform work. It also returns a weight for each active member that indicates the member's current capacity. (Be aware that the list can include members that are not represented by the location alias.) The requester uses this information to connect to the member or members with the most capacity that are also associated with the location alias.

**Restriction:** DB2 Connect does not support member-specific access in an SNA environment.

**Two-phase commit resynchronization**
If two-phase commit resynchronization is necessary, due to previous communication or system failure, the resynchronization request is sent to the member that is involved in the failure. DB2 performs resynchronization asynchronously, so that requesters are not prevented from establishing new sessions with other members.

**RACF PassTicket limitation**
If you use RACF PassTickets, you can define only one location alias. The generic LU name is used to generate a valid PassTicket for the data sharing group. Generation of a valid PassTicket for individual members or sets of members in a group is not possible.

**Group-generic access**
With group-generic access, a requester uses the generic LU name of the group to make an initial connection to any member of the group. VTAM then chooses one of the group members and establishes a session with that member on behalf of the requester. The requester uses the LU name of the chosen member when it makes subsequent requests.

VTAM provides workload balancing at the session level by allocating each connection request to the member with the fewest number of sessions at the time of connection. VTAM determines which member has the most capacity based on either the number of each member's active DDF sessions or the result of a user-written VTAM or z/OS Workload Manager exit routine.

After a connection is established between a requester and a member, all requests from the same requester that are made during the session are directed to the same member. VTAM does not reassess a member's capacity at the time of subsequent requests. Only after all connections between the requester and member are closed, can VTAM choose a different member to process future requests from the requester.

**Exception:** If the connection between the requester and the member is enabled for two-phase commit processing, VTAM continues to direct all requests from this requester to the same member. The mapping between the requester LU and the member LU is preserved until the DB2 command RESET GENERICLU is issued.
If a requester is active at the time of a communication or system failure, VTAM handles the reconnection in one of two ways:

- If the pre-failure session was enabled for two-phase commit support, VTAM reconnects the requester to the same member. If that member is unavailable, a communication error is returned.
- If the pre-failure session was not enabled for two-phase commit support, VTAM can connect the requester to any member.

Because VTAM balances the workload based on the number of currently established sessions rather than on true capacity, you should carefully evaluate this access method before implementing it. If the number of sessions is an accurate reflection of your workload, this method can be a good choice. It has the added benefit of being relatively easy to set up.

**Single-member access**

Single-member access is the same method used to access DB2 in non-data sharing environments. For detailed information about connecting to DB2 by using this method, consult [DB2 Installation Guide](#).

With single-member access, a requester uses the LU name in its TCP/IP configuration to connect to a specific member of the group. The requester sends all connection and query requests to this same member.

**DB2 Connect**: Configure DB2 Connect to use single-member access by disabling its Sysplex support. If you do not disable Sysplex support, DB2 Connect uses DRDA workload balancing by default to allocate requests among the members of the group.

**DB2 UDB for z/OS**: Configure DB2 UDB for z/OS to use single-member access by creating and using a location alias that represents only one member of the data sharing group. Otherwise, DB2 UDB for z/OS uses DRDA workload balancing by default to allocate requests among the members of the group.

**Recommendation**: Do not use single-member access for the following reasons:
- It is dependent on the specified member being operational.
- It provides no workload balancing for requesters that cannot perform DRDA workload balancing.

**Restriction**: Version 8 of DB2 Connect does not support single-member access.

**Setting up DB2 UDB for z/OS as a requester**

When DB2 acts as a requester, it can connect applications that run on the system to remote database servers including DB2 UDB for z/OS, DB2 UDB for OS/390, DB2 UDB for iSeries, and DB2 for VSE & VM.

In its role as a requester, DB2 UDB for z/OS accepts DB2 location names and translates them into SNA NETID.LUNAMEs. It uses the communications database (CDB) to register location names and their corresponding network parameters. The data that is stored in the CDB enables DB2 to pass the required information when making distributed database requests over SNA connections.

Much of the processing in a distributed database environment requires the exchange of messages with other locations in the network. For this processing to work correctly, you must:
1. Define the DB2 UDB for z/OS requester to the local VTAM system.
   See DB2 Installation Guide for detailed instructions on defining the DB2 requester to the local VTAM system.

2. Identify the remote data sharing groups to which applications can connect.

**Identifying remote data sharing groups**

When an application requests data from a remote subsystem, the DB2 UDB for z/OS requester searches the CDB for information about the remote group. Recall that DB2 uses the CDB to store information about how to communicate with remote groups, and that the DDF uses the CDB to map DB2 location names to LU names.

Member-specific access requires:
- Location alias entries in the SYSIBM.LOCATIONS table
- Conversational security requirements entries in the SYSIBM.LUNAMES table
- Location alias member entries in the SYSIBM.LULIST table

Group-generic access requires:
- A DB2 location name entry in the SYSIBM.LOCATIONS table
- A conversational security requirements entry in the SYSIBM.LUNAMES table

Single-member access requires:
- A location alias entry in the SYSIBM.LOCATIONS table
- A conversational security requirements entry in the SYSIBM.LUNAMES table
- A location alias member entry in the SYSIBM.LULIST table

The following sections provide more detail about each of these tables.

**SYSIBM.LOCATIONS**: SYSIBM.LOCATIONS maps the location names in connection requests to the VTAM LU names of remote systems and, if necessary, transaction program names (TPNs). SYSIBM.LOCATIONS must contain at least one row for each remote group, depending on the access method that is used.
- For member-specific access, the LOCATION column of each row contains a location alias that identifies one, several, or all members of the group.
- For group-generic access, the LOCATION column of the row contains the group's DB2 location name and the LINKNAME column of the row contains that group's generic LU name.
- For single-member access, the LOCATION column of the row contains a location alias that identifies one member of the group.

**SYSIBM.LUNAMES**: SYSIBM.LUNAMES maps location names to the LU names of remote systems. It also maps location names to the security and mode requirements for conversations with those systems. SYSIBM.LUNAMES must contain at least one row for each remote group, depending on the access method that is used.
- For member-specific access, the LINKNAME column of each row contains a link name.
- For group-generic access, the LUNAME column of the row contains the group's generic LU name.
- For single-member access, the LINKNAME column of the row contains a link name.
The GENERIC column of SYSIBM.LUNAMES is for the specific use of data sharing
group members that act as requesters. Its value indicates whether a requester uses
the group’s generic LU name or its own real LU name when identifying itself to a
remote subsystem. If you want a requester to use the group’s generic LU name,
specify a value of Y; if you want a requester to use its own LU name, specify a
value of N.

Important: The value of the GENERIC column is ignored if the DB2 GENERIC
LUNAME parameter on installation panel DSNTIPR is blank. In this case, DB2
requesters are identified to remote systems by their own LU names.

Be aware that only one member of a data sharing group can access a remote
system by using the group’s generic LU name. If one member of a group is already
connected to a remote system and is using the group’s generic LU name,
subsequent connections to that remote system by other members of the same
group use the requesters’ own LU names.

SYSIBM.LULIST: SYSIBM.LULIST supports member-specific access to a remote
data sharing group by enabling you to associate location aliases with one or more
members of the group. SYSIBM.LULIST must contain a row for every member that
is associated with a location alias. For members that are associated with multiple
location aliases, insert multiple rows.

• For member-specific access, the LUNAME column of each row contains a
  member’s LU name.
• For single-member access, the LUNAME column of the row contains the
  member’s LU name.

Example: Assume that a remote data sharing group has six members. You might
have two location aliases, one of which is associated with three of the members
and the other of which is associated with two of the members. In this example,
you insert five rows into the SYSIBM.LULIST table, three rows for the members
that are associated with the first location alias and two rows for the members that
are associated with the second location alias.

Sending requests: When sending a request, DB2 uses the value in the
LINKNAME column of the SYSIBM.LOCATIONS table to determine which
network protocol to use.

• If DB2 finds that same value in the LINKNAME column of the
  SYSIBM.IP NAMES table, it uses TCP/IP.
• If DB2 finds that same value in the LUNAME column of the SYSIBM.LUNAMES
  table, it uses SNA.
• If DB2 finds that same value in both SYSIBM.IP NAMES and
  SYSIBM.LUNAMES, it uses TCP/IP.

Updating the communications database: You can update the CDB while the DDF
is active. Changes to SYSIBM.LOCATIONS, SYSIBM.LUNAMES, and
SYSIBM.LULIST take effect in the following manner:

• If the DDF has not yet tried to communicate with a particular remote group,
  updates take effect when the DDF attempts to communicate with that location.
• If the DDF has already attempted to communicate with a particular remote
  group, updates take effect the next time the DDF is started.

Updates do not affect existing conversations; existing conversations continue to
operate as if the updates had not occurred.
Configuring data sharing groups as servers

This section describes how to configure a DB2 UDB for z/OS data sharing group as a server.

If requesters will use DB2 private protocol to access the group, you must identify the DB2 location names of those requesters. This is the only requirement for configuring a data sharing group for member-specific access. For additional requirements for configuring a data sharing group for group-generic access, see "Configuring data sharing groups for group-generic access."

DB2 private protocol is supported only between requesters and servers that are both DB2 subsystems or data sharing groups. When a server receives a request from a DB2 requester that is using DB2 private protocol, the server validates the requester’s DB2 location name. For each DB2 requester that is using DB2 private protocol, specify the requester’s DB2 location name in the LOCATION column of the server’s SYSIBM.LOCATIONS table.

Example: Assuming that the DB2 location names of two DB2 requesters that are using DB2 private protocol are DB2Y and DB2Z, an excerpt of the server’s SYSIBM.LOCATIONS table would look similar to Table 30.

Table 30. Identifying the DB2 location names of DB2 requesters that use DB2 private protocol in a server’s SYSIBM.LOCATIONS table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>LINKNAME</th>
<th>TPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2Z</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

General-use Programming Interface

Use the following SQL statements to populate SYSIBM.LOCATIONS with the DB2 location names of DB2 requesters that use DB2 private protocol:

1. This statement inserts the DB2 location names of DB2 requesters that use DB2 private protocol into the SYSIBM.LOCATIONS table:
   
   ```sql
   INSERT INTO SYSIBM.LOCATIONS (LOCATION) 
   VALUES ('DB2Y');
   INSERT INTO SYSIBM.LOCATIONS (LOCATION) 
   VALUES ('DB2Z');
   ```


Configuring data sharing groups for group-generic access

This section describes the tasks that are specific to configuring a DB2 server for group-generic access. Some of the tasks are relevant for making requests, and some of the tasks are relevant for handling requests.

Enabling group-generic access to a data sharing group consists of the following tasks:

1. Including information in the coupling facility for support of VTAM generic resources (the ISTGENERIC structure)

   For more information about using VTAM’s generic resources, see VTAM for MVS/ESA Network Implementation Guide. To calculate storage for the ISTGENERIC structure, you also need information from IBM eServer zSeries Processor Resource/System Manager Planning Guide.
2. Defining a generic LU name for the group
3. Configuring the members to use the group’s generic LU name when identifying themselves to remote subsystems
4. Identifying the generic LU names of requesting data sharing groups
5. Identifying the LU names of individual requesters, such as DB2 Connect and members of requesting data sharing groups

**Defining a generic LU name for the group:** Define a generic LU name for the data sharing group by specifying the name in the DB2 GENERIC LUNAME parameter on installation panel DSNTIPR for each member. A generic LU name can be one to eight characters in length, and it is stored in each member’s bootstrap data set (BSDS).

**Using the originating member’s LU name:** Using the LU name of the originating member of the group as the generic LU name, and changing the originating member’s LU name can be useful. This is particularly true when the originating member is already acting as a server for applications that will migrate to the data sharing group. By configuring the group to use the originating member’s LU name, you avoid needing to make extensive changes to the CDBs of DB2 Connect requesters and to the CDBs of DB2 UDB for z/OS requesters.

To change the LU name of the originating member of the group, you must change both the LU name value in the member’s VTAM APPL statement and the LUNAME value in the member’s bootstrap data set (BSDS).

For example, assume that a group is formed whose originating member is already defined to the requesters that are shown in Figure 24. Until you define a generic LU name when you enable group-generic access, requesters can access DB2A’s data only through the originating member (LUDB2A). If the originating member is unable to handle requests, requesters are unable to access DB2A’s data.

![Diagram](Image)

*Figure 24. Example configuration before enabling group-generic access. Access is limited to a single member of the data sharing group.*
As shown in Figure 25 by enabling group-generic access and by using the LU name of the originating member as the generic LU name, requesters can, with minimal change, access DB2A’s data from any member of the group.

Figure 25. Example configuration after enabling group-generic access. All members of the group can potentially handle requests.

**Configuring members to use the group’s generic LU name:** Specify a value of Y in the GENERIC column of each row of the SYSIBM.LUNAMES table that corresponds to a remote subsystem with which the data sharing group communicates. The GENERIC column of the SYSIBM.LUNAMES table is for the specific use of group members that act as requesters. Its value indicates whether a member uses the group’s generic LU name or its own real LU name when identifying itself to a remote subsystem. The remote subsystem must be able to recognize the generic LU name.

**Important:** The value of the GENERIC column is ignored if the DB2 GENERIC LUNAME parameter on installation panel DSNTIPR is blank.

Be aware that only one member of a data sharing group can access a remote subsystem by using the group’s generic LU name. If one member of a group is already connected to a remote subsystem and is using the group’s generic LU name, subsequent connections to that remote subsystem by other members of the same group use the requesters’ own LU names. If the remote subsystem only accepts generic LU names, all requests from the members of the requesting group must be routed through one member.

**Clarification:** The GENERIC column does not determine whether a member uses group-generic or member-specific access. The rows, or lack of rows, in SYSIBM.LULIST determine whether a member uses group-generic or member-specific access.

If the remote subsystem starts CNOS processing first, VTAM uses the name with which the remote subsystem connected, whether this is the real LU name or the generic LU name. Because this behavior is not always predictable, the subsystem...
that is handling requests from the data sharing group should be able to accept either the generic LU name or the real LU name when group-generic processing is used.

**Example:** Assume that the LU names of two group members are LUDB2NY and LUDB2LA. An excerpt of the SYSIBM.LUNAMES table would look similar to Table 31.

Table 31. SYSIBM.LUNAMES table of a requesting data sharing group that uses group-generic access. Not all columns are shown.

<table>
<thead>
<tr>
<th>LUNAME</th>
<th>GENERIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUDB2NY</td>
<td>Y</td>
</tr>
<tr>
<td>LUDB2LA</td>
<td>Y</td>
</tr>
</tbody>
</table>

General-use Programming Interface

Use the following SQL statements to update the SYSIBM.LUNAMES table to specify that members use their group’s generic LU name when identifying themselves to remote data sharing groups:

```
UPDATE SYSIBM.LUNAMES
SET GENERIC='Y'
WHERE LUNAME='LUDB2NY';

UPDATE SYSIBM.LUNAMES
SET GENERIC='Y'
WHERE LUNAME='LUDB2LA';
```

End of General-use Programming Interface

Identifying the generic LU names of requesting data sharing groups: Specify the generic LU names of requesting data sharing groups in the LUNAME column of the server’s SYSIBM.LUNAMES table.

**Example:** Assume that the generic LU names of two groups are LUDSG1 and LUDSG2. An excerpt of the server’s SYSIBM.LUNAMES table would look similar to Table 32.

Table 32. Generic LU names of requesting data sharing groups in a DB2 server’s SYSIBM.LUNAMES table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LUNAME</th>
<th>GENERIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUDSG1</td>
<td></td>
</tr>
<tr>
<td>LUDSG2</td>
<td></td>
</tr>
</tbody>
</table>

General-use Programming Interface

Use the following SQL statements to populate SYSIBM.LUNAMES with the generic LU names of requesting data sharing groups:

1. This statement inserts the generic LU names of requesting data sharing groups into the SYSIBM.LUNAMES table:

```
INSERT INTO SYSIBM.LUNAMES (LUNAME) VALUES ('LUDSG1');
```

```
INSERT INTO SYSIBM.LUNAMES (LUNAME) VALUES ('LUDSG2');
```
Identifying the LU names of requesters: Specify the LU names of requesters in the LUNAME column of the server's SYSIBM.LUNAMES table. Individual requesters include DB2 Connect, DB2 UDB for z/OS, and the members of requesting data sharing groups.

Example: Assume that the LU names of the members of one requesting data sharing group are LUMEM1, LUMEM2, and LUMEM3, and that the LU names of the members of another requesting data sharing group are LUMEMA and LUMEMB. An excerpt of the server's SYSIBM.LUNAMES table would look similar to Table 33.

Table 33. LU names of requesters in a DB2 server's SYSIBM.LUNAMES table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LUNAME</th>
<th>GENERIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUMEM1</td>
<td></td>
</tr>
<tr>
<td>LUMEM2</td>
<td></td>
</tr>
<tr>
<td>LUMEM3</td>
<td></td>
</tr>
<tr>
<td>LUMEMA</td>
<td></td>
</tr>
<tr>
<td>LUMEMB</td>
<td></td>
</tr>
</tbody>
</table>

---

General-use Programming Interface

Use the following SQL statements to populate SYSIBM.LUNAMES with the LU names of requesters:

1. This statement inserts the LU names of requesters into the SYSIBM.LUNAMES table:

   ```sql
   INSERT INTO SYSIBM.LUNAMES (LUNAME) VALUES ('LUMEM1');
   INSERT INTO SYSIBM.LUNAMES (LUNAME) VALUES ('LUMEM2');
   INSERT INTO SYSIBM.LUNAMES (LUNAME) VALUES ('LUMEM3');
   INSERT INTO SYSIBM.LUNAMES (LUNAME) VALUES ('LUMEMA');
   INSERT INTO SYSIBM.LUNAMES (LUNAME) VALUES ('LUMEMB');
   ```

---

Tip: Using the default row. Instead of explicitly identifying the generic LU names of requesting data sharing groups and the real LU names of individual requesters, you can use the default row in SYSIBM.LUNAMES. This row is created by the DSNTIJSG installation job, and it enables access by any requester that is not specifically defined by another row in the table. (You can change the values in this row any time before starting the DDF.) The default row looks like that in Table 34 on page 137.
Table 34. Default row of a DB2 server’s SYSIBM.LUNAMES table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LUNAME</th>
<th>GENERIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>(8 blanks)</td>
<td></td>
</tr>
</tbody>
</table>

Connecting distributed partners in an SNA network

This section shows you how to configure DB2 Connect requesters and DB2 requesters to access remote data sharing groups in SNA networks. For information about how to configure other types of requesters, consult the product documentation for your specific requester.

This section includes:

- “Configuring a DB2 requester to use member-specific access”
- “Configuring a DB2 Connect requester to use group-generic access” on page 139
- “Configuring a DB2 requester to use group-generic access” on page 140
- “Switching from group-generic to member-specific access” on page 141

Configuring a DB2 requester to use member-specific access

This section describes how to update the CDB of a DB2 requester to use member-specific access.

To enable member-specific access to a remote data sharing group:

1. Define one or more location aliases that identify different sets of members.
   Insert each location alias into the LOCATION column of the requester’s SYSIBM.LOCATIONS table.
   If you use RACF PassTickets, define only one location alias.

2. Map each location alias to the security and mode requirements for conversations with each set of members.
   Insert the same link names into the LINKNAME columns of the requester’s SYSIBM.LOCATIONS and SYSIBM.LUNAMES tables. Insert the appropriate security and mode requirements for your site.
   If you use RACF PassTickets, specify the group’s generic LU name as the value of the LINKNAME columns.

3. Map each location alias to the LU names of the data sharing members in that set.
   For each member of a set, insert the LU name of the member into the LUNAME column of the requester’s SYSIBM.LULIST table.

4. Define location aliases on the server (remote data sharing group).
   For each location alias on the requester’s side, you must define a location alias at the server by using the DSNJU003 (change log inventory) utility. See “Using the DSNJU003 utility to update the BDS$” on page 142 for more information about the DSNJU003 utility.

Example of member-specific access: This example provides sample SQL statements for enabling member-specific access. It also shows the results of those statements in the form of table excerpts. It assumes that a remote data sharing group exists with a DB2 location name of DB2A and three members with names LUDB2AR, LUDB2B, and LUDB2C. Location aliases that are to be defined are DB2B (with members LUDB2AR and LUDB2B), DB2C (with members LUDB2B and LUDB2C), and DB2D (with member LUDB2AR).
SQL statements:

---

**General-use Programming Interface**

Use the following SQL statements to update the CDB of a DB2 UDB for z/OS requester to use member-specific access:

1. These statements define location aliases that identify different sets of members:

   ```sql
   INSERT INTO SYSIBM.LOCATIONS (LOCATION) VALUES ('DB2B');
   INSERT INTO SYSIBM.LOCATIONS (LOCATION) VALUES ('DB2C');
   INSERT INTO SYSIBM.LOCATIONS (LOCATION) VALUES ('DB2D');
   ```

2. These statements map each location alias to the security and mode requirements for conversations with each set of members:

   ```sql
   UPDATE SYSIBM.LOCATIONS 
   SET LINKNAME='ALIASB' 
   WHERE LOCATION='DB2B';
   INSERT INTO SYSIBM.LUNAMES (LUNAME) VALUES ('ALIASB');
   UPDATE SYSIBM.LOCATIONS 
   SET LINKNAME='ALIASC' 
   WHERE LOCATION='DB2C';
   INSERT INTO SYSIBM.LUNAMES (LUNAME) VALUES ('ALIASC');
   UPDATE SYSIBM.LOCATIONS 
   SET LINKNAME='ALIASD' 
   WHERE LOCATION='DB2D';
   INSERT INTO SYSIBM.LUNAMES (LUNAME) VALUES ('ALIASD');
   ```

3. These statements map each location alias to the LU names of the data sharing members in that set:

   ```sql
   INSERT INTO SYSIBM.LULIST (LINKNAME, LUNAME) VALUES ('ALIASB', 'LUDB2AR');
   INSERT INTO SYSIBM.LULIST (LINKNAME, LUNAME) VALUES ('ALIASC', 'LUDB2B');
   INSERT INTO SYSIBM.LULIST (LINKNAME, LUNAME) VALUES ('ALIASC', 'LUDB2C');
   INSERT INTO SYSIBM.LULIST (LINKNAME, LUNAME) VALUES ('ALIASD', 'LUDB2AR');
   ```

---

End of General-use Programming Interface

---

Table excerpts:

An excerpt of the SYSIBM.LOCATIONS table would look like Table 35.

Table 35. Location aliases for a remote data sharing group in a DB2 requester's SYSIBM.LOCATIONS table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>LINKNAME</th>
<th>TPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2B</td>
<td>ALIASB</td>
<td></td>
</tr>
<tr>
<td>DB2C</td>
<td>ALIASC</td>
<td></td>
</tr>
</tbody>
</table>
Table 35. Location aliases for a remote data sharing group in a DB2 requester's SYSIBM.LOCATIONS table (continued). Not all columns are shown.

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>LINKNAME</th>
<th>TPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2D</td>
<td>ALIASD</td>
<td></td>
</tr>
</tbody>
</table>

An excerpt of the SYSIBM.LUNAMES table would look like Table 36.

Table 36. Location aliases in a DB2 requester's SYSIBM.LUNAMES table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LUNAME</th>
<th>GENERIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASB</td>
<td></td>
</tr>
<tr>
<td>ALIASC</td>
<td></td>
</tr>
<tr>
<td>ALIASD</td>
<td></td>
</tr>
</tbody>
</table>

An excerpt of the SYSIBM.LULIST table would look like Table 37.

Table 37. LU names of the members that are associated with each location alias in a DB2 requester's SYSIBM.LULIST table. Not all columns are shown.

<table>
<thead>
<tr>
<th>LINKNAME</th>
<th>LUNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIASB</td>
<td>LUD2BAR</td>
</tr>
<tr>
<td>ALIASB</td>
<td>LUD2B</td>
</tr>
<tr>
<td>ALIASC</td>
<td>LUD2B</td>
</tr>
<tr>
<td>ALIASC</td>
<td>LUD2C</td>
</tr>
<tr>
<td>ALIASD</td>
<td>LUD2AR</td>
</tr>
</tbody>
</table>

**Configuring a DB2 Connect requester to use group-generic access**

This section outlines the most important steps in configuring a DB2 Connect requester to use group-generic access. For detailed information about these steps, and for complete information about DB2 Connect, refer to *IBM DB2 Connect User’s Guide*.

1. Use the Configuration Assistant to update the database directories that DB2 Connect uses to manage database connection information.

   See *IBM DB2 Connect User’s Guide* for detailed information about updating database directories.

   **Important:** You must enable Sysplex support for the data sharing group by specifying the SYSPLEX parameter in the parameter string of the target database name in the DCS directory.

2. If you want to enable applications to update data in multiple remote database servers with guaranteed integrity, use the DB2 Control Center to enable and test multisite updates.


3. Bind the DB2 Connect utilities, and any applications that were developed using embedded SQL, to the databases with which they will operate.

   See *IBM DB2 Connect User’s Guide* for detailed bind instructions.
Configuring a DB2 requester to use group-generic access

This section describes how to update the CDB of a DB2 requester to use group-generic access.

To enable group-generic access to a remote data sharing group:

1. Identify the generic LU name of the remote group.
   Insert the group’s generic LU name into the LUNAME column of the requester’s SYSIBM.LUNAMES table. Do not include in this table the real LU names of the members of the group.

2. Map the generic LU name to the group’s DB2 location name.
   Insert the generic LU name and the DB2 location name of the group into the LINKNAME and LOCATION columns of the requester’s SYSIBM.LOCATIONS table.

Example of group-generic access: This example provides sample SQL statements for enabling group-generic access. It also shows the results of those statements in the form of table excerpts. It assumes that a remote data sharing group exists with a DB2 location name of DB2R and a generic LU name of LUDSGA.

SQL statements:

<table>
<thead>
<tr>
<th>General-use Programming Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the following SQL statements to update the CDB of a DB2 UDB for z/OS requester to use group-generic access:</td>
</tr>
<tr>
<td>1. This statement identifies the group’s generic LU name:</td>
</tr>
<tr>
<td>INSERT INTO SYSIBM.LUNAMES (LUNAME, GENERIC)</td>
</tr>
<tr>
<td>VALUES ('LUDSGA', 'Y');</td>
</tr>
<tr>
<td>2. This statement maps the group’s generic LU name to the group's DB2 location name.</td>
</tr>
<tr>
<td>INSERT INTO SYSIBM.LOCATIONS (LOCATION, LINKNAME)</td>
</tr>
<tr>
<td>VALUES ('DB2R', 'LUDSGA');</td>
</tr>
</tbody>
</table>

| End of General-use Programming Interface |

Table excerpts:

An excerpt of the SYSIBM.LUNAMES table would look like Table 38:

<p>| Table 38. Generic LU name of a remote data sharing group in a DB2 requester's SYSIBM.LUNAMES table. Not all columns are shown. |</p>
<table>
<thead>
<tr>
<th>LUNAME</th>
<th>GENERIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUDSGA</td>
<td>Y</td>
</tr>
</tbody>
</table>

An excerpt of the SYSIBM.LOCATIONS table would look similar to Table 39:

<p>| Table 39. Mapping the generic LU name of a remote data sharing group to its DB2 location name in a DB2 requester's SYSIBM.LOCATIONS table. Not all columns are shown. |</p>
<table>
<thead>
<tr>
<th>LOCATION</th>
<th>LINKNAME</th>
<th>TPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2R</td>
<td>LUDSGA</td>
<td></td>
</tr>
</tbody>
</table>
Switching from group-generic to member-specific access

Member-specific access provides several benefits over group-generic access, including better workload balancing and two-phase commit support. However, before a requester that was using group-generic access can take advantage of these benefits by switching to member-specific access, you need to break any affinity that might exist between the requester and the server. This section describes how to break the affinity between systems.

If a requester uses group-generic access to connect to a data sharing group with two-phase commit support enabled on both subsystems, VTAM records information in the coupling facility about which member of the group is involved in the communication. This information is required to ensure that future VTAM sessions are directed to the same group member, thus providing access to the correct member log for resolution of indoubt threads. When switching the requester from using group-generic access to member-specific access, you must break this affinity between the systems.

To break the affinity between systems, do the following:

1. Shut down the network connections between the requester and the data sharing group.
   Some ways to do this include:
   • Using the VTAM command VARY NET,INACT,ID=luname
   • Entering the DB2 command STOP DDF
2. From an active member of the data sharing group, issue the RESET GENERICLU command.
   Issue this command from the member with the VTAM affinity to the requester whose information is being deleted.
   • If the requester is configured to pass its own, real LU name to the data sharing group, netid is the net ID of the requester, and luname is the real LU name of the requester.
   • If the requester is configured to pass its generic LU name, netid is the net ID of the requesting data sharing group, and luname is the generic LU name of the requesting data sharing group.
   See DB2 Command Reference for detailed syntax and usage information for the RESET GENERICLU command.
3. Change the requester’s CDB to ensure that the requester uses member-specific access from this point forward.
   • Populate the requester’s SYSIBM.LULIST table, as described in “Configuring a DB2 requester to use member-specific access” on page 137.
     Make sure that the GENERIC column of the requester’s SYSIBM.LUNAMES table contains a value of N for the row that is associated with the data sharing group.
   • Delete existing generic LU names from the SYSIBM.LUNAMES table.
4. Re-enable the network connections between the requester and the data sharing group.
   Unlike group-generic access, which uses generic LU names, member-specific access uses real LU names. When switching the requester from using group-generic access to member-specific access, remember to delete existing generic LU names.
Preventing a member from processing requests

Transparency to end users, you can prevent one or more members of a data sharing group from handling DDF requests while still letting those members make DDF requests.

You can prevent a member from handling DDF requests in one of two ways:

- Set the MAX REMOTE ACTIVE option of installation panel DSNTIPE to zero for that member.
- Dynamically update the DSN6SYSP macro’s MAXDBAT parameter value to zero for that member.

The MAX REMOTE ACTIVE option and the MAXDBAT parameter specify the maximum number of database access threads (DBATs) that can be active concurrently. By setting the value of either to zero, you restrict DDF server activity on the affected member. Subsequent connection requests are directed to those members whose MAX REMOTE ACTIVE option and MAXDBAT parameter value are greater than zero. Any work that is already in progress by the affected member continues, but new requests are directed to other members of the data sharing group.

The effects of setting the value of the MAX REMOTE ACTIVE option or the MAXDBAT parameter to zero are:

- DDF does not register the member with z/OS Workload Manager (WLM) for member-specific access. If the member is already registered with WLM, the member is unregistered. The member can continue to use WLM for setting priorities on work, but the member’s name is not included on the list that is returned on initial connections to requesters that use member-specific access. Therefore, DDF requests are never routed to that member.
- In an SNA network, DDF does not register the member’s LU name with the group’s generic LU name during DDF startup.
- In a TCP/IP network, DDF does not listen on the DRDA port.

Using the DSNJU003 utility to update the BDS

You can use the DSNJU003 (change log inventory) utility to update the following information related to DB2 data sharing. This information is stored in the bootstrap data sets (BSDSs) of members of data sharing groups:

- DRDA port (PORT=port)

  The DRDA port is the TCP/IP port number that is used by the DDF to accept incoming connection requests. 446 is the recommended DRDA port.

  If you change this value for one member of the group, you must change it for all members. DB2 requires that all members of a data sharing group use the same, well-known port number to receive incoming connection requests. After making the change, you must stop and restart the DDF for the change to take effect.

- Generic LU name (GENERIC=gluname)

  The generic LU name represents all the members of a data sharing group, in either an SNA network or a TCP/IP network.

  If you change this value for one member of the group, you must change it for all members. All members of a data sharing group must specify the same name.

  After making the change, you must stop and restart the DDF for the change to take effect.

- Location aliases (ALIAS=alias-name:alias-port)
Location aliases can represent one, several, or all members of a data sharing group. Location aliases are useful in several ways:

- Member-specific access requires that you identify to the server (remote data sharing group) each location alias that is defined by a requester. Doing so enables a server to recognize itself as the intended recipient of connection requests that specify location aliases instead of the server's location name.

- After adding a new member to a data sharing group, you can create an alias for the member's old (subsystem) DB2 location name that maps to its new (group) DB2 location name. Doing so enables applications that are coded to connect to the member's old DB2 location name to continue to work.

- Location aliases enable you to define subsets of data sharing group members. Subsetting gives you the ability to limit the members to which DRDA requesters can connect.

- **Resynchronization port (REPORT=port)**
  The resynchronization port is the TCP/IP port number that is used by the DDF to accept incoming DRDA two-phase commit resynchronization requests.
  DB2 requires that each member of a data sharing group in a TCP/IP network have a resynchronization port number that is unique within the Parallel Sysplex. In the event of a failure, this unique port number allows a requester to reconnect to the correct member so that units of work that require two-phase commit can be resolved.

**Important:** Before updating any of the information related to DB2 data sharing, have all members check for the existence of indoubt threads. If any indoubt threads exist, resolve them before making any updates. To check for indoubt threads, use the DISPLAY THREAD command. See *DB2 Command Reference* for detailed information about this command.

See Part 2 of *DB2 Utility Guide and Reference* for complete information about the DSNJU003 utility.
Chapter 5. Operating with data sharing

Most data sharing operations are accomplished by using commands to DB2 UDB for z/OS. This chapter describes the following information:

- “Entering commands”
- “Effect of data sharing on sequence number caching” on page 146
- “Starting DB2” on page 147
- “Stopping DB2” on page 147
- “States of connections and structures after stopping DB2” on page 147
- “Submitting work to be processed” on page 148
- “Monitoring the group” on page 150
- “Controlling connections to remote systems” on page 156
- “Establishing the logging environment” on page 158
- “Recovering data” on page 160
- “Restarting DB2 after termination” on page 183
- “Starting duplexing for a structure” on page 196
- “Stopping duplexing for a structure” on page 196
- “Shutting down the coupling facility” on page 197

Entering commands

This section describes the following information:

- “Routing commands”
- “Command scope”
- “Entering commands from an application program” on page 146
- “Authorizing commands” on page 146
- “Receiving messages” on page 146

Routing commands

You can control operations on an individual member of a data sharing group from any z/OS console by entering commands prefixed with the appropriate command prefix. For example, assuming you chose -DB1A as the command prefix for member DB1A, you can start a DB2 statistics trace on that member by entering this command at any z/OS console in the Parallel Sysplex:

-DB1A START TRACE (STAT)

Command routing requires that the command prefix scope is registered as S or X on the IEFSSNxx parmlib member. For specifications of command prefixes, see “Registering command prefixes and member group attachment name” on page 61.

You can also control operations on certain objects by using commands or command options that affect an entire group. These, also, can be entered from any z/OS console. For example, assuming that DB1A is active, you can start database XYZ by entering this command at any z/OS console in the Parallel Sysplex:

-DB1A START DATABASE (XYZ)

Command scope

The breadth of a command’s impact is called the scope of that command.
Many commands that are used in a data sharing environment affect only the member for which they are issued. For example, a STOP DB2 command stops only the member identified by the command prefix. Such commands have member scope.

Other commands have group scope because they affect an object in such a way that all members of the group are affected. For example, a STOP DATABASE command, issued from any member of the group, stops that database for all members of the group. See [DB2 Command Reference](#) for information about the scope of each command.

**Entering commands from an application program**

You can enter commands from an application program attached to a DB2 subsystem through any of the attachment facilities: IMS, CICS, TSO, CAF, and RRSAF. Commands that are entered in this way are executed by the DB2 subsystem to which the application program is attached. The application cannot send a command to a different DB2 subsystem.

**Authorizing commands**

Data sharing does not introduce any new techniques for establishing and checking authorization IDs. Because all members of a data sharing group share the DB2 catalog, any ID has the same privileges and authorities on every member.

It is your responsibility to use the same connection or sign-on exit routines on every member of the data sharing group to avoid authorization anomalies.

**Receiving messages**

You can receive messages from all members at a single console. Hence, a message must include a member identifier as well as a message identifier. The member's command prefix appears in messages to identify the source of a message.

---

### Effect of data sharing on sequence number caching

DB2 always assigns sequence numbers in order of request. Members in a data sharing environment can use the CREATE SEQUENCE statement with the CACHE option to request a “chunk” of sequence numbers. These numbers become the members local cache. Since each member assigns cache from its local chunk, numbers may not be assigned in order across the data sharing group. Each value is guaranteed to be unique.

For example, the following CREATE SEQUENCE statement results in sequence SEQ1 being defined as a block of 20 cache values numbered 1 through 20:

```
CREATE SEQUENCE SEQ1 START WITH 1 INCREMENT BY 1 CACHE 20
```

Because each data sharing group member gets its own sequenced block of cache values, all values assigned to that member are from that block. If member DB2A requests the first cache value for SEQ1, it is assigned value 1. If member DB2B requests the next cache value, it is assigned value 21. When member DB2A requests another cache value, it is assigned value 2, and when member DB2B requests another cache value, it is assigned value 22. When a member’s block of cache values is exhausted, the next available block of cache values is allocated. For more information about the CREATE SEQUENCE statement, see [DB2 SQL Reference](#).
Starting DB2

To start members of a data sharing group, you must enter a START DB2 command for each member of the group. If this is the first startup for the group, you must start the originating member first.

Impact of command prefix scope: If DB2 is installed with a command prefix scope of STARTED (the default and recommended value), you must issue the START DB2 command from the z/OS system on which you want to start DB2 or you must route the command to that z/OS system. For example, to route the START DB2 command to the z/OS system on which you want to start DB1A, issue the following command:

ROUTE MVS1,-DB1A START DB2

After DB2 is started, you can issue all other commands from any z/OS system in the Parallel Sysplex, and the commands are routed to the appropriate member.

Stopping DB2

Stop individual members of a data sharing group by using the STOP DB2 command as described in Part 3 of [DB2 Command Reference]. Consider specifying CASTOUT(NO) when you stop an individual member of a data sharing group for maintenance. This option speeds up shutdown because DB2 bypasses castout and associated cleanup processing in the group buffer pools.

States of connections and structures after stopping DB2

When DB2 allocates its coupling facility structures, it specifies a disposition for the structures and for connections to the structures after a normal or abnormal termination. When you display the structures, you can see different states for the connections and structures based on how the disposition is defined and whether DB2 was stopped normally or shut down abnormally.

Normal shutdown

Table 40 summarizes the information that you see after a normal termination.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Connections</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCA</td>
<td>None</td>
<td>Allocated</td>
</tr>
<tr>
<td>Lock</td>
<td>Failed-persistent</td>
<td>Allocated</td>
</tr>
<tr>
<td>Group buffer pools</td>
<td>None with CASTOUT(YES); Failed-persistent with CASTOUT(NO)</td>
<td>Unallocated with CASTOUT(YES); Allocated with CASTOUT(NO)</td>
</tr>
</tbody>
</table>

Notes:

1. If a member has no retained locks, its failed-persistent connection to the lock structure is removed when it shuts down. If this is the last member to shut down, the connection remains in a failed-persistent state.
2. If castout failure occurs during shutdown, group buffer pool connections show as failed-persistent, even though DB2 terminates normally.
Abnormal shutdown

Table 41 summarizes the information that you see after an abnormal termination.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Connections</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCA</td>
<td>None</td>
<td>Allocated</td>
</tr>
<tr>
<td>Lock</td>
<td>Failed-persistent 1</td>
<td>Allocated</td>
</tr>
<tr>
<td>Group buffer pools</td>
<td>Failed-persistent</td>
<td>Allocated</td>
</tr>
</tbody>
</table>

Note:
1. If a member has no retained locks, its failed-persistent connection to the lock structure is removed when it shuts down. If this is the last member to shut down, the connection remains in a failed-persistent state.

Submitting work to be processed

The methods you use to submit work need not change for data sharing. However, you might find it to your advantage to use the group attachment name to direct jobs. This section describes more about the group attachment name and how it works.

Using the group attachment name

Utilities and applications that use TSO, batch, DL/I batch, the RRSAF, the CAF, or DB2I (DB2 Interactive) to connect to DB2 have two methods for specifying the member to which they want to connect. The first method is to specify the name of the subsystem. The second method is to use the group attachment name, which acts as a generic name for all the members of a data sharing group. The group attachment name can be used in place of the name of the DB2 subsystem that runs on the z/OS system from which the job was submitted.

By using the group attachment name, the utility or application does not need to be sensitive to a particular member, which makes it easier to move jobs around the Parallel Sysplex as needed. The utility or application connects to the first active group member that it finds. However, as described in "Specifying the group attachment name” on page 61, when the name of a subsystem is the same as the group attachment name, the utility or application attempts to connect to the member first. If that member is not started and group attachment name processing is not disabled, the utility or application attempts to connect to the next group member that is active. (NOGROUP is the group override and is available for CAF and RRSAF.)

You specify the group attachment name at DB2 installation on the DSNTIPK installation panel, which places the name in the IEFSSNx.x member and in the DSNHDECP load module for the group. The group attachment name appears in the output for the command DISPLAY GROUP.

If you do not explicitly specify a subsystem name or group attachment name, DB2 uses DSN as the name of the intended subsystem. As with any application program, make sure you are accessing the set of DB2 libraries with the correct DSNHDECP programming defaults.

For more information about submitting applications, see DB2 Application Programming and SQL Guide.
Running CICS and IMS applications

You can specify the group attachment name when running CICS and IMS applications for data sharing. CICS and IMS applications must be aware of the particular member to which they are attached so they can resolve indoubt units of recovery in the event of a failure. IMS Version 7 or later allows IMS dependent regions to use the DB2 group attachment name to direct jobs to a data sharing group. CICS Transaction Server Version 2.2 or later allows you to use the CICS RESYNCMEMBER=YES option to handle indoubt units of work. See Part 4 (Volume 1) of DB2 Administration Guide for more information about running applications.

Submitting online utility jobs

When you submit a utility job, you must specify either the group attachment name or the name of the member to which the utility is to attach. For example, if you specify the group attachment name, the EXEC statement might look like the following statement:

```
//stepname EXEC PGM=DSNUTILB,PARM='group-attach-name,[uid],[utproc]'  
```

Establishing affinity

If you do not use the group attachment name, the utility job must run on the z/OS system where the specified DB2 subsystem is running. Ensure that the utility job runs on the appropriate z/OS system. You must use one of several z/OS installation-specific statements to make sure this happens. These include:

- For JES2 multi-access spool (MAS) systems, insert the following statement into the utility JCL:

  ```
  //*JOBPARM SYSAFF=cccc  
  ```

  where `cccc` is the JES2 name. You can specify an asterisk (`SYSAFF=*`) to indicate that the job should run on the system from which it was submitted.

- For JES3 systems, insert the following statement into the utility JCL:

  ```
  //*MAIN SYSTEM=main-name  
  ```

  where `main-name` is the JES3 name.

The preceding JCL statements are described in z/OS MVS JCL Reference. Your installation might have other mechanisms for controlling where batch jobs run, such as the use of job classes.

Stopping and restarting utilities

In a data sharing environment, you can use the TERM UTILITY command to stop the execution of an active utility only from the member on which the utility is active. You can then use this same command from any active member of the data sharing group to release the resources that are associated with the stopped utility. If a member fails while a utility is executing, you must restart DB2 on either the same or another z/OS system before stopping the utility. For remote site recovery from a disaster at the local site, you can stop utilities that were active at the local site from any restarted member of the group at the remote site.

**Recommendation:** Define all work data sets that are used by utilities on shared disks. Doing so allows you to restart a utility on any member, since all members are able to access all required data sets. Use the same utility ID (UID) to restart the utility. UIDs are unique within a data sharing group. If a member fails while a utility is executing, you must restart DB2 on either the same or another z/OS system before restarting the utility.
Altering utilities
In a data sharing environment, you can use the ALTER UTILITY command to alter the REORG utility only from the member on which the utility is active. This is true for non-data sharing environments as well.

Submitting stand-alone utility jobs
DB2 stand-alone utilities (such as DSN1COPY) run as z/OS jobs that have no direct connection to DB2 services. Therefore, a data sharing group has no indication that one of these utilities is running.

In a data sharing environment, if a table space has inter-DB2 read/write interest, its most recently updated pages might be in the coupling facility and a stand-alone utility might not be running with current data. If it is important that the data is in a consistent state, you must stop or quiesce the table space. Also, the data must not be in the recovery pending (RECP) or group buffer pool recovery pending (GRECP) state nor have any logical page list entries. Use the DISPLAY DATABASE command with the RESTRICT option to find out if there are exception statuses for a given table space or index.

Monitoring the group
This section describes the commands you can use to perform the following tasks:
- "Obtaining information about the group"
- "Obtaining information about structures and policies" on page 151
- "Obtaining information about group buffer pools" on page 153
- "Monitoring databases" on page 154
- "Determining the data sharing member on which SQL statements run" on page 156

The section on monitoring databases includes information about the logical page list and how to clear entries from that list. It also includes information about detecting retained locks.

Obtaining information about the group

**General-use Programming Interface**

To obtain general information about all members of a particular group, use the DISPLAY GROUP command, shown here:

```
-DB1A DISPLAY GROUP
```

The command can be issued from any active member of the group, and it displays the output that is shown in Figure 26 on page 151.
Figure 26 shows the following information:

- The DB2 group name and group release level
- The group attachment name
- The member names and release levels
- The command prefix for each member
- The status of each member (ACTIVE, QUIESCED with or without additional conditions, or FAILED)
- The DB2 release level of each member
- The name of the z/OS system where each member is running (or was last running, in cases where the member is not active)
- The names of the IRLM subsystems to which members are connected. For more information about the IRLM data sharing group, use the z/OS command 

For more information about the lock table and the list of modify locks, see “Avoiding false contention” on page 223.

**Parallel query information:** To see the COORDINATOR and ASSISTANT subsystem parameters for all active members of the group, use the DETAIL option of DISPLAY GROUP. See Figure 36 on page 208 for an example.

---

**End of General-use Programming Interface**

---

**Obtaining information about structures and policies**

Use the z/OS command D XCF,STR to display information about coupling facility structures and policy information. For more information about the D XCF,STR command, see z/OS MVS System Commands.
Displaying all structures

The following command displays summary information about all structures:

```
D XCF,STR
```

The command produces the output that is shown in Figure 27:

```
04/09/1999 15:57:47 DUPLEXING REBUILD NEW STRUCTURE
04/09/1999 15:29:43 DUPLEXING REBUILD OLD STRUCTURE
04/09/1999 15:26:16 ALLOCATED
04/09/1999 15:26:13 ALLOCATED
04/09/1999 15:11:09 ALLOCATED
```

**Figure 27. Output from D XCF,STR command**

Displaying information about specific structures

You can also display more detailed information about specific structures. The following command displays information about the duplexed group buffer pool GBP0 for group DSNCAT:

```
D XCF,STR,STRNAME=DSNCAT_GBP0
```
The command produces the output that is shown in Figure 28:

```
D XCF,STR,STRNAME=DSNCAT GBP0
IXC360I 11.13.38 DISPLAY XCF
STRNAME: DSNCAT GBP0
STATUS: REASON SPECIFIED WITH REBUILD START:
       OPERATOR INITIATED
       DUPLICATING REBUILD
       REBUILD PHASE: DUPLEX ESTABLISHED
POLICY SIZE : 32768 K
POLICY INITSIZE: 5000 K
REBUILD PERCENT: N/A
DUPLICATE : ALLOWED
PREFERENCE LIST: LF01 CACHE01
EXCLUSION LIST IS EMPTY

DUPLICATING REBUILD NEW STRUCTURE
-----------------------------------------------
ALLOCATION TIME: 04/12/1999 11:13:31
CFNAME : CACHE01
COUPLING FACILITY: SIMDEV.IBM.EN.ND020000000
       PARTITION: 0 CPCID: 00
ACTUAL SIZE : 5120 K
STORAGE INCREMENT SIZE: 256 K
VERSION : B2162049 D1E56F02
DISPOSITION : DELETE
ACCESS TIME : 0
MAX CONNECTIONS: 32
# CONNECTIONS : 2

DUPLICATING REBUILD OLD STRUCTURE
-----------------------------------------------
ALLOCATION TIME: 04/12/1999 11:12:51
CFNAME : LF01
COUPLING FACILITY: SIMDEV.IBM.EN.ND010000000
       PARTITION: 0 CPCID: 00
ACTUAL SIZE : 5120 K
STORAGE INCREMENT SIZE: 256 K
VERSION : B2162023 A5B3CB06
ACCESS TIME : 0
MAX CONNECTIONS: 32
# CONNECTIONS : 2

CONNECTION NAME ID VERSION SYSNAME JOBNAME ASID STATE
--------------- ------- --------- --------- ---------- ----- ----------
DB2_V71A 02 00020001 UTEC277 V71ADB01 002F ACTIVE NEW,OLD
DB2_V71B 01 00010001 UTEC277 V71BDB01 0033 ACTIVE NEW,OLD
```

Figure 28. Output from D XCF,STR,STRNAME command

Obtaining information about group buffer pools

To obtain information about group buffer pools, you can use the z/OS command D XCF,STR as described in “Displaying information about specific structures” on page 152. However, DB2 also provides a DISPLAY GROUPBUFFERPOOL command that is useful for displaying statistical information about group buffer pool use.

Depending on the options you choose for the command, the display output contains the following information:

- A list of all connections to the group buffer pools. For duplexed group buffer pools, only one set of connections exists for both instances of the group buffer pool. For example, if there are three connections to duplexed structure GBP0, there are just three connections, not six connections.
• Statistical reports on group buffer pool use, either by a specific member or by the whole group. Some statistical information is also available for the secondary allocation of a duplexed group buffer pool.

See “Using the DISPLAY GROUPBUFFERPOOL command” on page 260 for more information about the DISPLAY GROUPBUFFERPOOL command.

**Monitoring databases**

The group buffer pool recovery pending (GRECP) and logical page list (LPL) statuses are specific to DB2 data sharing. These statuses can appear on the output from the DISPLAY DATABASE command.

**GRECP**

“Group buffer pool recovery pending.” The group buffer pool was lost, and the changes that are recorded in the log must be applied to the page set. When a page set is placed in the GRECP state, DB2 sets the starting point for the merge log scan to the log record sequence number (LRSN) of the last complete group buffer pool checkpoint.

DB2 automatically recovers GRECP page sets when the group buffer pool is defined with AUTOREC (YES) and at least one member was connected when the failure occurred.

**LPL**

“Logical page list.” Some pages were not read from, or written to, the group buffer pool because of some failure, such as a channel failure between the group buffer pool and the processor. Or perhaps pages could not be read from, or written to, disk because of a transient disk problem.

In Version 8 and higher, DB2 attempts to automatically recover logical page list page sets. For page sets or partitions that have LPL or GRECP status and that are not automatically recovered, either start the page set or partition using the START DATABASE command with SPACENAM and ACCESS (RW) or (RO), or run the RECOVER utility. If any table or index space that is required to confirm START DATABASE command authority is unavailable, INSTALL SYSADM might be required to issue the command. See DB2 Command Reference for more information about authorization requirements. For more information about removing LPL status, see “Recovering pages on the logical page list” on page 168.

**Obtaining information about pages in error**

The logical page list contains a list of pages (or a page range) that could not be read or written for some reason, such as transient disk read and write problems that can be fixed without redefining new disk tracks or volumes.

Specific to data sharing, the LPL also contains pages that could not be read or written for “must-complete” operations, such as a commit or a restart, because of some problem with the coupling facility. For example, pages can be added if there is a channel failure to the coupling facility or disk, or if locks are held by a failed subsystem, thus disallowing access to the desired page.

The logical page list is kept in the SCA and is thus accessible to all members of the group.

If an application tries to read data from a page that is on the logical page list, it receives a “resource unavailable” SQLCODE. In order to be accessible, pages in the
logical page list must first have their logged changes applied to the page set.

### General-use Programming Interface

To verify the existence of logical page list entries, issue the DISPLAY DATABASE command. The LPL option of DISPLAY DATABASE can then be used to see the specific list of pages:

```
-DB1A DIS DB(DSNDB01) SPACENAM(*) LIMIT(*) LPL ONLY
```

Output similar to the following is produced:

```
DSNT360I -DB1A
******************************************************************************
DSNT360I -DB1A * DISPLAY DATABASE SUMMARY
* GLOBAL LPL
DSNT360I -DB1A
******************************************************************************
DSNT362I -DB1A DATABASE = DSNDB01 STATUS = RW
DBD LENGTH = 8000
DSNT397I -DB1A
NAME TYPE PART STATUS LPL PAGES
------- ----- ----- -------------- --------------
DBD01 TS RW,LPL,GRECP 0000001,000004,00000C,000010
------- 000039-00003C
SYSLGRNX TS RW,LPL,GRECP 000000-FFFFFFFF
******************************************************************************
DSN9022I -DB1A DSN00IS 'DISPLAY DATABASE' NORMAL COMPLETION
```

Automatic LPL recovery is attempted when pages are added to the logical page list. Automatic LPL recovery is also performed when you start the table space, index, or partition by using the START DATABASE command with ACCESS(RW) or ACCESS(RO).

### Physical read and write errors

In some previous versions of DB2, physical read and write errors were recorded in an error page range. This is still the case; however, if a read or write problem is of undetermined cause, the error is first recorded in the logical page list. If recovery from the logical page list is unsuccessful, the error is then recorded in the error page range.

### Obtaining information about locks held during DB2 failure

When a lock is used to allow an object to be changed (called a modify lock), the lock is kept in a list in the coupling facility lock structure to allow for recovery in case a member fails. If a member fails, modify locks become retained locks, which means that they are held until the failed subsystem is restarted. For more information about retained locks, see “Active and retained locks” on page 183.

To determine if there are retained locks, use the DISPLAY DATABASE command with the LOCKS option as shown here:

```
-DB1A DISPLAY DATABASE(TESTDB) LOCKS ONLY
```

You can tell if a lock is retained if there is an R in the LOCKINFO field of the report.

```
NAME TYPE PART STATUS CONNID CORRID LOCKINFO
------- ----- ----- -------------- -------------- -------------
TBS43 TS 01 RW MEMBER NAME DB2A R-IX,PP
```
Removing retained locks: A normal restart of DB2 resolves and removes retained locks held by that member with the full data integrity control that DB2 restart provides. However, if you cannot restart DB2 and the failed member has retained locks that are severely affecting transactions on other members, consider the following actions:

- Defer the restart processing of the objects that have retained locks.
  When you defer restart processing, the pages that locks are protecting are placed in the logical page list. Those pages are still inaccessible. However, this approach has the advantage of removing any retained page set P-locks, which have the potential of locking out access to an entire page set. See Part 4 (Volume 1) of DB2 Administration Guide for more information about deferred restart.
- Cold start the failed member.
  This approach causes DB2 to purge the retained locks, but data integrity is not protected. When the locks are released after the cold start, DB2 looks at data whose status is unclear. See “Restarting a member with conditions” on page 194 for more information about how to do this.
- Use the command, MODIFY irlmproc,PURGE,dbname.
  Like a cold start, this command causes DB2 to purge the retained locks and with this method, data integrity is not protected. See Part 3 of DB2 Command Reference for more information about MODIFY irlmproc,PURGE.
- Restart the failed member in light mode (restart light).
  Restart light is not recommended for a restart in place. It is intended for a cross-system restart in the event of a failed z/OS to quickly recover retained locks. Restart light enables DB2 to restart with a minimal storage footprint and then terminate normally after the locks are released. For more information about restart light, see “Restart light” on page 185.

Determining the data sharing member on which SQL statements run

Use the special register CURRENT MEMBER to determine the member of a data sharing group on which SQL statements execute. The data type of CURRENT MEMBER is CHAR(8). If necessary, the member name is padded on the right with blanks so that its length is 8 bytes. The value of CURRENT MEMBER is a string of blanks when the application process is connected to a DB2 subsystem that is not a member of a data sharing group.

Example: To set the host variable MEM to the name of the current member, use one of the following statements:
EXEC SQL SET :MEM = CURRENT MEMBER;
EXEC VALUES (CURRENT MEMBER) INTO :MEM

Controlling connections to remote systems

This section describes how controlling DDF connections is changed for data sharing. The following topics are described:
- “Starting and stopping DDF” on page 157
- “Monitoring connections to remote systems” on page 157
- “Resetting generic LU information” on page 157
Starting and stopping DDF

General-use Programming Interface

You control the distributed data facility (DDF) on a member basis, not on a
group-wide basis. This gives you more granular control over DDF processing. For
example, assume that you want to devote member DB1A to batch processing for
some period of time without disrupting other connections. You can enter the
following command to disallow any further distributed connections from coming
into this member:
-DB1A STOP DDF MODE(QUIESCE)

To stop all DDF processing for the group, you need to issue the STOP DDF
command for every member of the group. For example, you might need to do this
when you change the SYSIBM.Locations table.

To allow for completion of CREATE, ALTER, DROP, GRANT, or REVOKE
operations from remote requesters, issue the STOP DDF MODE(SUSPEND)
command.

End of General-use Programming Interface

Monitoring connections to remote systems

General-use Programming Interface

The DISPLAY THREAD command can display information about all threads in the
data sharing group. The DISPLAY LOCATION command shows thread
information only for the member on which it is issued.

If a data sharing group is defined to have a generic LU, you must use a member's
real LU name for luwid if you are requesting information by luwid (logical unit of
work ID).

If the DISPLAY THREAD is being related to the global transactions coordinated by
xid, DISPLAY THREAD shows xid information. The thread is managed by an XA
transaction manager, such as Websphere, which identifies the transaction with an
xid. The xid is provided to allow correlation with the XA transaction manager. DB2
uses both the logical unit of work identifier, luwid, and the XA transaction
identifier, xid, to coordinate and recover transactions.

When a remote DB2 subsystem issues a DISPLAY LOCATION command to obtain
information about connections to a data sharing group, the output displays
information about every LU at that location.

For more information about the DISPLAY THREAD and DISPLAY LOCATION
commands, see Part 3 of DB2 Command Reference

End of General-use Programming Interface

Resetting generic LU information

If you use a generic LU name to connect to a member of a data sharing group that
is using two-phase commit, VTAM permanently records information in the
coupling facility about which member of the group was involved in the
communication. This permanently recorded information is required to guarantee
that future VTAM sessions are always directed to the same member, making it possible to provide access to the correct member log for resolution of indoubt threads.

At times, you might need to break that affinity between the member and the other system. You would need to do this, for example, if you want to use member-specific access, or if you want to remove a member from the data sharing group.

---

**General-use Programming Interface**

To break this affinity, issue the `RESET GENERICLU` command from an active member of the data sharing group. You must issue this command from the member with the VTAM affinity to the particular LU. Here is an example of a command that removes information about USIBMSTODB22 from DB1A:

```
-D81A RESET GENERICLU(LUD822)
```

Take great care when using this command because it can potentially cause the specified partner LUs to connect to different members on later sessions. This can cause operational problems if indoubt threads exist at a partner LU when this command is issued.

---

For more information about using the `RESET GENERICLU` command, see Part 3 of *DB2 Command Reference*.

---

**Establishing the logging environment**

In a data sharing environment, the member subsystems still maintain separate recovery logs. Each manages its own active and archive log data sets and records those in its own bootstrap data set (BSDS). The shared communications area (SCA) in the coupling facility contains information about all members’ BSDSs and log data sets. In addition, every member’s BSDS contains information about other members’ BSDS and log data sets, in case the SCA is not available.

*Figure 29* illustrates a typical logging environment.

---

*Figure 29. Member BSDSs and logs in the data sharing environment*
The impact of archiving logs in a data sharing group

In data sharing, the DB2 RECOVER utility needs log records from every member that has changed the object needing to be recovered. More information about how RECOVER uses these logs is described in “How recovery works in a data sharing group” on page 161. If the logs are archived, the impact on RECOVER depends on how the log data sets are archived:

- Archive to disk without DFSMShsm to migrate the data sets from disk to tape.
  No major impact on performance is experienced for this type of archive. But you need enough disk space to hold archive logs, and the disk devices must be shared (accessible) by all members in a data sharing group. Because DFSMShsm or its equivalent is not used, you must manage disk space carefully to avoid running out of space.
- Archive to disk with DFSMShsm.
  DFSMShsm can do automatic space and data availability management among storage devices in a system. DFSMShsm can migrate the archive on disk to less expensive storage (such as tape), and recall the archive back to disk when needed.

Using DFSMShsm, a particular RECOVER job needs only one tape unit to recall migrated archive data sets. If the archive data sets have been migrated, recovery time might be adversely affected, because the recalls of the migrated archive data sets are done one at a time from the member running the RECOVER job. For example, a RECOVER job started on DB1A might need log data sets from DB1A, DB2A, and DB3A. DB1A sends the recall requests to DFSMShsm one at a time for the tapes needed for recovery.

- Archive to tape.
  The RECOVER job needs at least one tape unit for each member whose archived log records are to be merged. (More might be needed if you run more than one recover at the same time for different partitions of a partitioned table space.) Therefore, do not archive logs from more than one system to the same tape.

**Recommendation:** For data sharing, avoid using tape archive logs for data recovery.

If you must archive to tape, ensure that the value for READ TAPE UNITS on installation panel DSNTIPA for each member is high enough to handle anticipated recovery work. For example, if you have eight members, each member should specify at least eight tape drives. You need more if you run more than one recovery job at the same time on a given member, or if multiple members run recovery jobs at the same time.

If there are not enough tape units to do the recovery, DB2 can possibly deadlock. If this happens, use the command SET ARCHIVE to increase the number of tape units that can be used.

Also, ensure that you specify 0 for the DEALLOC PERIOD parameter on installation DSNTIPA to avoid making an archive tape inaccessible to other members of the data sharing group. (If you intend to run all RECOVER jobs from a single member, this suggestion does not apply.)

How to avoid using the archive log

A recovery cycle for a table space is defined by how often its image copy is taken. A RECOVER job needs the following copies and records:

- The latest image copy
- The optional incremental copies
- The log records since the last incremental or image copy
• Archived log records, if all of the log records since the last incremental image copy are not still in active log data sets

Keeping all of the log records since the last incremental image copy in the active log data sets is to your advantage, because reading log records from the active log is much faster than reading from archive logs, even if those archives are on disk.

Several ways exist to minimize the need to use the archive log:
• Increase the total active log space.
  The total amount of active log space is the number of active log data sets multiplied by its size. Currently, DB2 limits the maximum number of active log data sets to 31. Because each member can have up to 31 active log data sets, the total number of active log data sets is effectively increased by the number of members in a data sharing group.
  The size of an active log data set is up to 2 GB but is usually limited by the size of a tape cartridge. Most installations prefer not to have an archived data set on more than one tape volume.

  Recommendation: Do not use tape compression for the DB2 archive log, because DB2 needs to read the log backwards for backout operations. Performance for backout can be severely degraded if tape compression is used.
  Tape compression is not DB2 data compression, which compresses the data portion of a DB2 log record. With DB2 data compression, the log record header is not compressed and causes no extra performance degradation for backward scans.
• Increase the frequency of incremental image copies.
  Because only the log records generated since the last incremental image copy are needed for recovery, the more often you make incremental image copies, the less chance there is that archive log records will be needed. Weigh this consideration against the time it takes to make the incremental image copies and the effects on SQL transactions.

  See “Preparing for faster recovery” on page 162 for more information about improving recovery performance.
• Make sure applications commit frequently.
  To avoid having to mount an archive log for backing out changes, ensure that applications are committing frequently. Consider using the UR CHECK FREQ parameter or the UR LOG WRITE CHECK parameter of installation panel DSNTIPL to help you track when applications are not committing as frequently as your site guidelines suggest.

Recovering data

This section describes the changes in data recovery that are required by data sharing, including data affected by the failure of the coupling facility or the structures within the coupling facility.

The procedures for data recovery are fundamentally the same for data sharing as for non-data-sharing environments. Data sharing involves one catalog, but many logs and BSDSs. In addition to disk and cache controllers, a new medium, the coupling facility, is introduced. This adds a possible point of failure and necessitates appropriate procedures for data recovery. In planning for data sharing, consider having more than one coupling facility. If an SCA or lock structure failure occurs, recovery for that structure can proceed automatically if a second coupling facility is available.
The following topics are described in this section:

- “How recovery works in a data sharing group”
- “Preparing for faster recovery” on page 162
- “Using the RECOVER utility” on page 163
- “System-level point-in-time recovery” on page 164
- “Recovering a data sharing group in case of a disaster” on page 164
- “Recovering pages on the logical page list” on page 168
- “Recovery from coupling facility failures” on page 168
- “Coupling facility recovery scenarios” on page 173
- “Resolution of Transaction Manager indoubt units of recovery” on page 181

How recovery works in a data sharing group

This section describes how the recovery process works for a shared object.

Determining the logs needed for recovery

Assume that three members of a data sharing group are making updates to table space TS1, as shown in Figure 30.

The following is the sequence of overlapping updates leading up to the time of recovery:

1. DB1A updates TS1 between Time 1 and 4 (SYSLGRNX record 1) with two updates (U1 and U4).
2. DB3A updates TS1 between Time 2 and 5 (SYSLGRNX record 2) with three updates (U2, U3, and U5).
3. DB2A updates TS1 between Time 6 and 13 (SYSLGRNX record 3) with three updates (U6, U8, and U13).
4. DB1A updates TS1 again between Time 7 and 11 (SYSLGRNX record 4) with three updates (U7, U10, and U11).
5. DB3A updates TS1 again between Time 9 and 15 (SYSLGRNX record 5) with four updates (U9, U12, U14 and U15).

Now, assume that you want to recover TS1 to time 9. The full image copy taken at T0 is used as the recovery base. All the SYSLGRNX records mentioned previously are selected to determine the log ranges of each system for the log scan. Updates U1 through U9 are applied in order.
Applying the log records

DB2 can access the logs of other DB2 subsystems in the group and merge them in sequence. The log record sequence number (LRSN) uniquely identifies the log records of a data sharing member. The LRSN is always incremented for log records that pertain to the same page. Duplicate LRSNs never exist for the same page, but LRSNs might be duplicated in log records of members on different pages. Figure 31 illustrates the structure of the log record.

![Log and LRSN structure](image)

**Figure 31. The log and LRSN in the data sharing environment.** During recovery, DB2 compares the LRSN in the log record with the LRSN in the data page to determine whether the log record must be applied to the data on disk.

The log record header contains an LRSN. The LRSN is a 6-byte value that is greater than or equal to the timestamp value truncated to 6 bytes. This value also appears in the page header. During recovery, DB2 compares the LRSN in the log record to the LRSN in the page header before applying changes to disk. If the LRSN in the log record is larger than the LRSN on the data page, the change is applied.

Preparing for faster recovery

This section describes two ways to increase your recovery performance: perform more frequent image copies and enable faster log apply.

Increase the frequency of image copies

One way to prepare for a faster recovery is to increase the frequency of image copies. You might want to limit this activity by determining which table spaces most need fast recovery. The following guideline is provided as a starting point to help you determine how often you must do incremental image copies. Refer to Part 2 of [DB2 Utility Guide and Reference](https://www.ibm.com) for information about using the COPY utility and all the associated ramifications with using incremental versus full image copies. As with a single subsystem, doing frequent image copies can help you avoid using the archive log for recovery.

Use the guideline below for each member of the data sharing group. Use the output of the DSNJU004 (print log map) utility for each member.

1. Find the starting timestamp of the active log data set with the lowest STARTRBA.
2. Find the ending timestamp of the active log data set with the highest ENDRBA.
3. Calculate the time interval:
   \[
   \text{time interval} = \text{end TIMESTAMP} - \text{start TIMESTAMP}
   \]
4. Calculate the interval at which to perform incremental image copies:
   \[
   \text{interval of copy} = \frac{\text{time interval} \times (n-1)}{n}
   \]
   where \( n \) is the number of active log data sets.
5. Take the smallest interval for the group and, to account for fluctuations of log activity, decrease the interval by 30%. (30% is an arbitrary figure; you might have to adjust this interval based on your system's workload.) This is the recommended interval for doing incremental image copies. If the interval is too small to be realistically accomplished, consider increasing the size or number of active log data sets. Periodically run the MERGECOPY utility with incremental image copies. The RECOVER utility attempts to mount tape drives for all the incremental image copies at the same time. If it runs out of tape drives, it switches to log apply. MERGECOPY merges what it can and then mounts more incremental image copies.

Enable fast log apply
Improve recovery times (and restart times) by providing enough storage for the DB2 fast log apply process. This process is able to sort log records so that pages that are to be applied to the same page or same set of pages are together. Then, using several log apply tasks, it can apply those records in parallel.

Provide storage for this process using the LOG APPLY STORAGE parameter of installation panel DSNITPL.

Using the RECOVER utility
Use the RECOVER utility to recover to currency or to a prior point in time. The details of RECOVER are described in Part 2 of [DB2 Utility Guide and Reference](#).

Recovery to currency
This process is used to recover from damaged data by restoring from a backup and applying all logs to the current time. The recovery process operates similarly in the data sharing and non-data-sharing environments. Image copies are restored and subsequently updated based on changes recorded in the logs. In the data sharing group, multiple member logs are read concurrently in log record sequence.

Point-in-time recovery
This process discards potentially corrupt data by restoring a database to a prior point of consistency. Corrupt data might result from a logical error. The following point-in-time recovery options are available:

**TORBA**  This option is used to recover to a point on the log defined by a receive byte address (RBA). In a data sharing environment, TORBA can only be used to recover to a point prior to defining the data sharing group.

**TOLOGPOINT**  This option is used to recover to a point on the log defined by a log record sequence number (LRSN). The TOLOGPOINT keyword must be used when you recover to a point on the log after the data sharing group was defined. However, you can also use TOLOGPOINT in a non-data-sharing environment.

The LRSN is a 6-byte hexadecimal number derived from a store clock timestamp. LRSNs are reported by the DSN1LOGP stand-alone utility.

**TOCOPY**  This option is used to recover data or indexes to the values contained in an image copy without subsequent application of log changes.
Successful recovery clears recovery pending conditions and brings data to a point of consistency. In a data sharing environment, all pages associated with the recovered data entity are removed from the group buffer pool and written to disk.

**System-level point-in-time recovery**

DB2 provides the following solutions for system-level recovery:

- Recovery to a specific point in time
  This point in time is either between two backup times or between the last backup time and the current time. After the appropriate volume copies are restored, the outstanding logs are applied to the databases to recover the data to the designated point in time.

- Recovery to the point in time of a backup
  This recovery restores the appropriate volume copies of the data and logs. The logs are used only to back out inflight transactions on restart.

- Remote site recovery from disaster at a local site
  This recovery is determined by what you keep at the remote site. For example, in addition to offsite tapes of backups, current logs might have been transmitted electronically. If so, the current logs can be applied to the databases after the data and logs are restored from the offsite tapes.

Use the BACKUP SYSTEM and RESTORE SYSTEM online utilities to perform system-level point-in-time recovery.

**BACKUP SYSTEM online utility**

This utility provides the following types of system copies:

- A data-only system backup, which contains only the databases
  The RESTORE SYSTEM online utility uses these backups to recover the system to an arbitrary point in time.

- A full system backup, which contains both logs and databases
  The RESTORE SYSTEM online utility uses these backups to recover the system to the point in time when the copy was taken by using normal DB2 restart recovery.

**RESTORE SYSTEM online utility**

This utility recovers a DB2 subsystem to a prior point in time by restoring volume copies that are provided by the BACKUP SYSTEM online utility with the DATA ONLY option. After restoring the data, you can use the RESTORE SYSTEM online utility to recover to an arbitrary point in time.

See the operation and recovery section of the [DB2 Administration Guide](#) for more information about system-level point-in-time recovery. See the [DB2 Utility Guide and Reference](#) for more information about the BACKUP SYSTEM and RESTORE SYSTEM online utilities.

**Recovering a data sharing group in case of a disaster**

This section presents an overview of how to recover a data sharing group at a remote site. To develop a procedure, you can use as a base the disaster recovery procedure documented in Part 4 (Volume 1) of the [DB2 Administration Guide](#) With a couple of exceptions, you must perform those steps for each member of the data sharing group. The following topics describe how to prepare for recovery of a data sharing group at a recovery site:

- "Configuring the recovery site" on page 165
- "What to send to the recovery site" on page 166
“Recovery procedure differences” on page 167 is the procedure you use to prepare the data sharing group at the recovery site for a group restart.

*Using a tracker site for disaster recovery:* As an alternative, you can set up the remote data sharing group as a tracker site. The advantage of a tracker site is that it dramatically reduces the amount of time needed for takeover should a disaster occur at the primary site. The disadvantage is that the tracker site must be dedicated to shadowing the primary site. You cannot use the tracker site to perform transactions of its own. See Part 4 (Volume 1) of *DB2 Administration Guide* for more information about setting up and using a tracker site.

**Configuring the recovery site**

The recovery site must have a data sharing group that is identical to the group at the local site. It must have the same name, the same number of members, and the members must have the same names as those at the local site. The CFRM policies at the recovery site must define the coupling facility structures with the same names as those at the local site (although the sizes can be different).

You can run the data sharing group on as few or as many *z/OS* systems as you want.

The hardware configuration can be different at the recovery site, as long as it supports data sharing. Conceptually, there are two ways to run the data sharing group at the recovery site. Each way has different advantages that can influence your choice:

- **Run a multi-system data sharing group.**
  
  The local site is most likely configured this way. You have a Parallel Sysplex containing many CPCs, *z/OS* systems, and DB2 subsystems. This configuration requires a coupling facility, the requisite coupling facility channels, and the Sysplex Timer.
  
  Using a multi-system data sharing group at the recovery site, you have the same availability and growth options that you have at the local site.

- **Run a single-system data sharing group.**
  
  In this configuration, you centralize all of your DB2 processing within a single, large processor, such as an IBM eServer™ zSeries 900 or 990. As Figure 32 on page 166 shows, you must *install* a multi-member data sharing group. After the group starts up, you shut down all but one of the members and access data through the remaining member.
With a single-system data sharing group, you lose the availability benefits of the Parallel Sysplex, but the group has fewer hardware requirements:

- The Sysplex Timer is not needed; you can use the CPC’s time-of-day clock.
- You can use any available coupling facility configuration for the recovery site system, including Integrated Coupling Facilities (ICFs).

With a single-system data sharing group, there is no longer inter-DB2 read/write interest, and the requirements for the coupling facility are as follows:
- A lock structure (which can be smaller)
- An SCA

Group buffer pools are not needed for running a single-system data sharing group. However, you do need to have at least small group buffer pools for the initial startup of the group. DB2 allocates them and uses them to do its damage assessment processing. When you are ready to do single-system data sharing, you can remove the group buffer pools by stopping all members and then restarting the member that is handling the workload at the disaster recovery site.

For more information about ICF, see IBM eServer zSeries Processor Resource/System Manager Planning Guide.

What to send to the recovery site
You must send to the recovery site the same information as for single-system remote recovery: logs and BSDSs, image copies, and so on. To prepare the logs for the remote site, you have three options:

- Use the command ARCHIVE LOG with the MODE (QUIESCE) option to ensure a point of consistency for each of the log data sets. If the quiesce is not successful, the command fails and the logs are not archived.

At the recovery site (just as for non-data-sharing disaster recovery), the ENDRBA value you use for restarting each member is the end RBA +1 of the latest archive log data set from each member in the data sharing group.

- Use the command ARCHIVE LOG SCOPE(GROUP). This version of the command does not ensure a point of consistency for all members’ logs, but the logs are archived on each of the active members of the data sharing group. You
can use the ENDLRSN option of the DSNJU003 (change log inventory) utility on the remote site to truncate all logs to the same point in time. To determine the truncation value, look at the print log map output from the latest copies of the archived BDS.

Another way to determine the truncation value is to ship the SYSLOG containing message DSNJ003I with your archive log data sets to the recovery site. This message is issued when archive log data sets are created (when you issue the ARCHIVE LOG command). The message contains the starting and ending LRSN and RBA values for the archive log data set. For example, the following messages appear when the command ARCHIVE LOG SCOPE(GROUP) is issued from one of the members at the local site:

```
DSNJ003I -DB1A DSNJ0FF3 FULL ARCHIVE LOG VOLUME
DSNAME=DSNC510.ARCHLOG1.A0000003, STARTRBA=000001C60000, ENDRBA=000001D4FFFF, STARTLRSN=ADFA208AA36C, ENDLRSN=AE3C45273A77, UNIT=SYSDA, COPY1VOL=SCR03, VOLSPAN=00, CATLG=YES

DSNJ003I -DB2A DSNJ0FF3 FULL ARCHIVE LOG VOLUME
DSNAME=DSNC518.ARCHLOG1.A0000001, STARTRBA=000000000000, ENDRBA=00000000DFFFF, STARTLRSN=ADFA00B70FB, ENDLRSN=AE3C452760D7, UNIT=SYSDA, COPY1VOL=SCR03, VOLSPAN=00, CATLG=YES
```

Compare the ending LRSN values for all members’ archive logs, and choose the lowest LRSN as the truncation point. For the two members shown previously, the lowest LRSN is AE3C45273A77. To get the last complete log record, you must subtract 1 from that value, so you would enter AE3C45273A76 as the ENDLRSN value in the CRESTART statement of the DSNJU003 utility for each of the members at the remote site. All log records with a higher LRSN value are discarded during the conditional restart.

- Use the command SET LOG SUSPEND if you are using the IBM RAMAC® Virtual Array (RVA) storage control with the peer-to-peer remote copy (PPRC) function or Enterprise Storage Server Flashcopy to create point-in-time backups of entire DB2 subsystems for faster recovery at a remote site. Using either of these methods to create a remote copy requires the suspension of logging activity, which prevents database updates. The SUSPEND option of the SET LOG command suspends logging and update activity until a subsequent SET LOG command with the RESUME option is issued. For more information about using RVA Snapshot or Enterprise Storage Server Flashcopy for remote site recovery, see Part 4 (Volume 1) of the DB2 Administration Guide.

**Important:** Make sure that all members of the group are active when you archive the logs. If you have a quiesced member whose logs are necessary for a recovery base at the disaster recovery site, you must start that member with ACCESS(MAINT) to archive its log.

For read-only members, DB2 periodically writes a log record to prevent those members from keeping the LRSN value too far back on the log.

**Recovery procedure differences**

The procedure for data sharing at the recovery site differs in that extra steps exist for cleaning out old information in the coupling facility. Old information exists in the coupling facility from any practice startups. In addition, you must prepare each subsystem (rather than just a single system) for a conditional restart. For the detailed procedure, see Part 4 (Volume 1) of the DB2 Administration Guide.
Recovering pages on the logical page list

DB2 attempts to automatically recover pages when they are added to the logical page list. Recovered pages are deleted from the logical page list when recovery processing successfully completes. In some situations—such as a channel failure between the group buffer pool and the processor, a transient disk problem, or DB2 in restart—the automatic LPL recovery processor cannot be initiated, or the recovery cannot complete. In these situations, the pages are kept in the logical page list. The following tasks show several ways to perform manual LPL recovery:

• Start the object with access (RW) or (RO). This command is valid even if the table space is already started. When you issue the command START DATABASE, you see message DSNI006I, indicating that LPL recovery has begun. Message DSNI022I might be issued periodically to give you the progress of the recovery. When recovery is complete, you see message DSNI021I.

• Run the RECOVER utility on the object. The only exception to this is when a logical partition of a type 2 nonpartitioned index has both LPL and RECP status. If you want to recover the logical partition using RECOVER INDEX with the PART keyword, you must first use the command START DATABASE to clear the logical page list pages.

• Run the LOAD utility with the REPLACE option on the object.

• Issue an SQL DROP statement for the object.

• Use the utility REPAIR SET with NORCVRPEND. This can leave your data in an inconsistent state.

• Use START DATABASE ACCESS(FORCE). This can leave your data in an inconsistent state.

None of the items in the preceding list work if retained locks are held on the object. You must restart any failed member that is holding those locks.

Recovery from coupling facility failures

 Failures of the coupling facility can be classified into two main groups:
• Structure failure
  A structure failure is a rare event in which structures are damaged in some way but the coupling facility continues to operate.
• Connectivity failures
  Connectivity failures can be caused by a problem with the attachment of the z/OS system to the coupling facility. They can also occur when the following types of failures occur:
  – Power failure that affects the coupling facility but leaves one or more z/OS systems running
  – Deactivation of the coupling facility partition
  – Failure of the coupling facility control code
  – Failure of the coupling facility CPC or LPAR

This section also includes information about allocation failure and problems caused by a lack of storage.

Preparing for structure and connectivity failures

If properly configured, DB2 and IRLM can recover very quickly and with very little disruption from any kind of coupling facility failure. If not properly configured, coupling facility failures can mean serious outages for users. Not
having a lock structure or SCA causes the entire data sharing group to come down abnormally. Group buffer pool failure can mean loss of availability for applications depending on the data in that group buffer pool.

Careful preparation can greatly reduce the impact of coupling facility outages to your users. To best prepare yourself for structure and connectivity failures, you must have the following:

- Group buffer pool duplexing enabled.
  Duplexing group buffer pools assures minimal impact to performance and can avoid hours of recovery time.
- Alternative coupling facility information provided on the preference list of each of the structures in the CFRM policy.
- For simplex coupling facility structures:
  - An active system failure management (SFM) policy with system weights specified.

  This is strongly recommended for simplex coupling facility structures. Descriptions of failure scenarios in this section assume that you have done this. If you have not, it is not possible to automatically rebuild simplex coupling facility structures. If the SCA and lock structure cannot be rebuilt, DB2 abnormally terminates the members affected by the loss of those structures, or the loss of connectivity to those structures. If the group buffer pool cannot be rebuilt, which is only attempted when a subset of members lose connectivity, those members disconnect from the group buffer pool.

- A REBUILDPERCENT value specified in the CFRM policy for all DB2-related structures

  In general, specifying a low REBUILDPERCENT value is recommended to allow for automatic rebuild when a member loses connectivity.

- Adequate storage in an alternate coupling facility to rebuild or reallocate structures as needed.

  For rebuild, z/OS uses the current size structure of the CFRM policy on the alternate coupling facility to allocate storage. If z/OS cannot allocate enough storage to rebuild the SCA or lock structure, the rebuild fails. If it cannot allocate enough storage for the group buffer pool, DB2 must write the changed pages to disk instead of rebuilding them into the alternate group buffer pool. For more information about how structure allocation works, see z/OS MVS Programming: Sysplex Services Guide.

For more information about planning, see “Coupling facility availability” on page 37.

Summary of failure scenarios
The tables in this section summarize connectivity and structure failure situations.

- “Connectivity failure to the SCA or lock structure” on page 170
- “Connectivity failure to non-duplexed group buffer pools” on page 170
- “Structure failures” on page 171
- “Summary of failure scenarios for duplexed group buffer pools” on page 172

For more information about specific recovery scenarios, see “Coupling facility recovery scenarios” on page 173.

**Connectivity failure to the SCA or lock structure**: Table 42 on page 170
summarizes what happens when there are connectivity failures to the SCA.
Table 42. Summary of connectivity failures to the SCA

<table>
<thead>
<tr>
<th>Active SFM policy?</th>
<th>Weighted loss &lt; REBUILDPERCENT?</th>
<th>DB2 response</th>
<th>Operational response</th>
</tr>
</thead>
</table>
| No                | Not applicable                  | Each affected member issues: DSN7501A 00F70600 | Options include:  
  • Fix problem  
  • Restart failed member on system that is connected to coupling facility  
  • Manually rebuild onto another coupling facility. |
|                   |                                 | DB2 comes down. Connection is deleted; structure remains allocated. |                       |
| Yes               | Yes                             | Each affected member issues: DSN7501A 00F70600 | Options include:  
  • Fix problem  
  • Restart failed member on system that is connected to coupling facility  
  • Manually rebuild onto another coupling facility. |
|                   |                                 | DB2 comes down. Connection is deleted; structure remains allocated. |                       |
| Yes               | No                              | Automatic rebuild. DSN7503I | None needed. |

Table 43 summarizes what happens when there are connectivity failures to the lock structure.

Table 43. Summary of connectivity failures to the lock structure

<table>
<thead>
<tr>
<th>Active SFM policy?</th>
<th>Weighted loss &lt; REBUILDPERCENT?</th>
<th>DB2 response</th>
<th>Operational response</th>
</tr>
</thead>
</table>
| No                | Not applicable                  | Each affected member issues: DXR136I 00E30105 | Options include:  
  • Fix problem  
  • Restart failed member on system that is connected to coupling facility  
  • Manually rebuild onto another coupling facility. |
|                   |                                 | DB2 comes down. Connection is failed-persistent; structure remains allocated. |                       |
| Yes               | Yes                             | Each affected member issues: DXR136I 00E30105 | Options include:  
  • Fix problem  
  • Restart failed member on system that is connected to coupling facility  
  • Manually rebuild onto another coupling facility. |
|                   |                                 | DB2 comes down. Connection is failed-persistent; structure remains allocated. |                       |
| Yes               | No                              | DXR143I      | None needed. |

Automatic rebuild. DXR146I

Connectivity failure to non-duplexed group buffer pools: Table 44 on page 171 summarizes what happens when there are connectivity failures to the group buffer pools.
### Table 44. Summary of connectivity failures for non-duplexed group buffer pools

<table>
<thead>
<tr>
<th>Connectivity lost from</th>
<th>Weighted loss &lt; REBUILDPERCENT?</th>
<th>DB2 response</th>
<th>Operational response</th>
</tr>
</thead>
<tbody>
<tr>
<td>100% of members connected to a group buffer pool that is defined with GBPCACHE(YES)</td>
<td>No</td>
<td>Each affected member issues: DSNB303E, DSNB228I</td>
<td>None needed if the group buffer pool is defined with AUTOREC(YES), and DB2 successfully recovers the page sets. Otherwise, enter START DATABASE commands.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Add pages to logical page list, if necessary. DSNB250E, DSNB314I* rsn=100%, DSNB304I*</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Damage assessment, GRECP page sets. DSNB320I, DSNB321I, DSNB353I, DSNI006, DSNI021, DSNB354I, DSNB305I*</td>
<td></td>
</tr>
<tr>
<td>100% of members connected to a group buffer pool that is defined with GBPCACHE(NO)</td>
<td>No</td>
<td>Automatic rebuild. DSNB331I, DSNB338I</td>
<td>None needed.</td>
</tr>
<tr>
<td>A subset of members connected to some or all group buffer pools.</td>
<td>Yes</td>
<td>Each affected member issues: DSNB303E, DSNB228I, DSNB313I rsn=LOSSCONN</td>
<td>Options include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quiesce applications that use the group buffer pool. Add pages to logical page list if necessary. DSNB250E, DSNB311I, DSNB312I</td>
<td>• Fix the problem.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Manually rebuild the structure onto another coupling facility.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Stop and restart DB2 on a system that is connected to the coupling facility.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A subset of members connected to some or all group buffer pools.</td>
<td>No</td>
<td>Automatic rebuild. DSNB331I, DSNB332I, DSNB333I, DSNB338I</td>
<td>None needed.</td>
</tr>
</tbody>
</table>

### Note:
1. Issued by the structure owner.

**Structure failures:** [Table 45 on page 172](#) summarizes what happens to each structure if there is a structure failure. This information is for non-duplexed group buffer pools. For more information, see:
- [“Problem: group buffer pool structure failure (no duplexing)” on page 176](#)
- [“Problem: lock structure failure” on page 177](#)
- [“Problem: SCA structure failure” on page 177](#)
Table 45. Summary of structure failures, by structure type

<table>
<thead>
<tr>
<th>Failed structure</th>
<th>DB2 response</th>
<th>Operational response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCA</td>
<td>DSN7502I</td>
<td>None needed.</td>
</tr>
<tr>
<td></td>
<td>Automatic rebuild</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSN7503I</td>
<td></td>
</tr>
<tr>
<td>Lock structure</td>
<td>DXR143I</td>
<td>None needed.</td>
</tr>
<tr>
<td></td>
<td>Automatic rebuild</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DXR146I</td>
<td></td>
</tr>
<tr>
<td>Group buffer pool</td>
<td>DSNB228I</td>
<td>None needed.</td>
</tr>
<tr>
<td>that is defined with</td>
<td>DSNB314I</td>
<td>rsn=STRFAIL</td>
</tr>
<tr>
<td>GBPCACHE(YES)</td>
<td>DSNB8250E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add pages to logical page list, if</td>
<td></td>
</tr>
<tr>
<td></td>
<td>necessary.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNB8250E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Damage assessment, GRECP page sets.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNB304I</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNB314I</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNB320I</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNB321I</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSN1006</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSN1021</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNB305I</td>
<td></td>
</tr>
<tr>
<td>Group buffer pool</td>
<td>DSNB331I</td>
<td>None needed.</td>
</tr>
<tr>
<td>that is defined with</td>
<td>DSNB338I</td>
<td></td>
</tr>
<tr>
<td>GBPCACHE NO</td>
<td>Automatic rebuild</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNB383I</td>
<td></td>
</tr>
</tbody>
</table>

Summary of failure scenarios for duplexed group buffer pools: For duplexed group buffer pools, a failure response is the same for both structure failures and for lost connectivity, as shown in Table 46.

Table 46. Summary of scenarios for both structure failure and lost connectivity for duplexed group buffer pools

<table>
<thead>
<tr>
<th>Failed structure</th>
<th>DB2 response</th>
<th>Operational response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>Switch to secondary in simplex mode</td>
<td>If the system did not automatically reduplex, correct the problem with the failed coupling facility. If you want to restart duplexing, use the z/OS SETXCF command.</td>
</tr>
<tr>
<td></td>
<td>DSNB744I</td>
<td>If DUPLEX(ENABLED) then reduplexing is attempted.</td>
</tr>
<tr>
<td></td>
<td>DSNB745I</td>
<td></td>
</tr>
<tr>
<td>Secondary</td>
<td>Revert to primary in simplex mode</td>
<td>If the system did not automatically reduplex, correct the problem. If you want to restart duplexing, use the SETXCF command.</td>
</tr>
<tr>
<td></td>
<td>DSNB743I</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNB745I</td>
<td></td>
</tr>
<tr>
<td>Both (structure failure or 100 %</td>
<td>Damage assessment, GRECP page sets.</td>
<td>None needed if the group buffer pool is defined with AUTOREC(YES) and DB2 successfully recovers the page set. Otherwise, enter START DATABASE commands.</td>
</tr>
<tr>
<td>LOSSCONN)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Coupling facility recovery scenarios

The following scenarios are described in this section:

- “Problem: all members have lost connectivity” on page 174
- “Problem: a subset of members have lost connectivity” on page 174
- “Problem: group buffer pool structure failure (no duplexing)” on page 176
- “Problem: group buffer pool structure failure (duplexing)” on page 176
- “Problem: lock structure failure” on page 177
- “Problem: SCA structure failure” on page 177
- “Problem: allocation failure of the group buffer pool” on page 177
- “Problem: storage shortage in the group buffer pool” on page 178
- “Problem: storage shortage in the SCA” on page 179
- “Problem: storage shortage in the lock structure” on page 180
- “Deallocating structures by force” on page 180

Because some problems might require you to deallocate structures by force, information is included about how to do that in “Deallocating structures by force” on page 180.

Problem: all members have lost connectivity

This scenario explains what you might see if there is a failure that is treated by z/OS and DB2 as a total loss of connectivity to the coupling facility. It assumes you have an active SFM policy.

Symptom: Some or all of the following messages appear, depending on which structures DB2 tries to access:

- DSNB303E -DB1A csect-name A LOSS OF CONNECTIVITY WAS DETECTED TO GROUP BUFFER POOL gpname.
- DSNB2281 csect-name GROUP BUFFER POOL gpname CANNOT BE ACCESSED FOR function.
- MVS IXLCACHE REASON CODE=reason.
- DSNB3141 csect-name DAMAGE ASSESSMENT TO BE TRIGGERED FOR GROUP BUFFER POOL gpname REASON=100%LCON
- DSNB250E csect-name A PAGE RANGE WAS ADDED TO THE LOGICAL PAGE LIST. DATABASE NAME = dbn SPACE NAME = spn DATA SET NUMBER = dsno PAGE RANGE = lowpg TO highpg START LRSN = startlrsn END LRSN = endlrsn START RBA = startrba
- DXR143I ilrmx REBUILDING LOCK STRUCTURE BECAUSE IT HAS FAILED OR AN IRLM LOST CONNECTION TO IT

System action: When all active members lose connectivity to the SCA or lock structure, these structures are rebuilt:

DSN7503I is issued for a successful rebuild of the SCA.
DXR146I is issued for a successful rebuild of the lock structure.

Important: If the lock structure or SCA cannot be rebuilt, the lost connectivity causes the members of the group to abend with abend code 00E30105 or 00F70600. This can happen, for example, if both coupling facilities are volatile and lose power. In this case, a group restart is required. Group buffer pools cannot be automatically recovered during group restart; you have to recover the group buffer pools with START DATABASE commands.
To avoid situations in which a group restart is necessary, put structures in nonvolatile coupling facility structures. See “Coupling facility volatility” on page 41 for more information.

For a loss of connectivity to the group buffer pools, applications that need access to group buffer pool data are rejected with a -904 SQL return code. Any of the following reason codes can be returned:

00C20204
00C20205
00C20220

- DB2 puts the group buffer pool in “damage assessment pending” status. The following message appears:

DSNB3041  -DB1A  csect-name GROUP BUFFER POOL
gbname WAS SET TO 'DAMAGE ASSESSMENT PENDING' STATUS

- DB2 adds entries to the logical page list, if necessary.

- DB2 marks the affected table spaces, indexes, or partitions as group buffer pool recovery pending (GRECP) (DSNB320I or DSNB321I), indicates that the group buffer pool is recovering page sets (DSNB353I), and initiates recovery for the page set (DSNI006I).

- As each page set is recovered, the castout owner for the page set issues DSNI021I. After the last page set is recovered, any member that has issued a DSNB353I now issues a DSNB354I.

- After damage assessment is complete, the structure owner issues DSNB305I.

The first new connection to the group buffer pool causes z/OS to reallocate the group buffer pool in the same or an alternate coupling facility as specified on the preference list in the CFRM policy.

**System programmer action:** You must fix the problem that is causing the loss of connectivity. If the problem is not an obvious one (such as a power failure), call IBM Software Support. After the problem is fixed, restart any failed members as quickly as possible to release retained locks.

- For the SCA or lock structure, if the automatic rebuild occurred normally, processing can continue while you wait for the coupling facility problem to be fixed.

- For lost connectivity to group buffer pools, DB2 automatically recovers data that was in any group buffer pool defined with AUTOREC(YES). If the automatic recovery is not successful or if any pages remain in the logical page list after recovery, issue START DATABASE commands for all affected page sets. You must issue separate START DATABASE commands in the following order.

1. DSNDB01
2. DSNDB06

If any table or index space that is required for authorization checking is unavailable, Installation SYSADM or Installation SYSOPR authority is required to issue the START DATABASE commands.

**Problem: a subset of members have lost connectivity**
This scenario describes what happens if one or more members lose connectivity to the coupling facility, but other members do not. This might happen if a link is detached between a system and the coupling facility. This scenario assumes that the group buffer pool, SCA, and lock structures are simplex, and that the combined weights of the systems that have lost connectivity are less than the weight that is required to trigger an automatic rebuild of the structure; that is, less than the value specified for the REBUILDPERCENT parameter of the CFRM policy.
When DB2 rebuilds the group buffer pool, it writes changed pages from the group buffer pool to the alternate structure specified in the CFRM policy. If DB2 determines that there is not enough space to hold the changed pages, it casts the pages out to disk instead.

Operator intervention is usually not required unless DB2 adds pages to the logical page list for some reason while the rebuild occurs. In this case, a START DATABASE command is needed to recover the pages on the logical page list.

**Symptom:** Some or all of the following messages appear, depending on which structures DB2 tries to access:

```
DSNB303E -DB1A csect-name A LOSS OF CONNECTIVITY WAS DETECTED TO GROUP BUFFER POOL gbpname.
DSNB301E -DB1A csect-name GROUP BUFFER POOL gbpname CANNOT BE CONNECTED
DB2 REASON CODE = reason1
MVS IXLCNN REASON CODE = xxxx0C06
DSNB228I csect-name GROUP BUFFER POOL gbpname CANNOT BE ACCESSSED FOR function
MVS IXLCACHE REASON CODE=reason
DSNB313I -DB1A csect-name GROUP BUFFER POOL gbpname TO BE DISCONNECTED
REASON=LOSSCONN
LOSSCONN PERCENTAGE=percentage
DSNB250E csect-name A PAGE RANGE WAS ADDED TO THE LOGICAL PAGE LIST
DATABASE NAME = dbn
SPACE NAME = spn
DATA SET NUMBER = dsno
PAGE RANGE = lowpg TO highpg
START LRSN = startlrsn
END LRSN = endlrsn
START RBA = startrba
DSNB311I csect-name DBNAME SPACE NAME spacename HAS PAGES IN THE LOGICAL PAGE LIST
DSNB312I csect-name DBNAME SPACE NAME spacename PARTITION part-number
HAS PAGES IN THE LOGICAL PAGE LIST
DSNB309I csect-name GROUP BUFFER POOL gbpname HAS BEEN DISCONNECTED WITH A REASON OF 'FAILURE'
DXR136I irlmx HAS DISCONNECTED FROM THE DATA SHARING GROUP
DSN7501A -DB1A csect-name SCA STRUCTURE sca-structure-name CONNECTIVITY FAILURE.
```

**System action:** For a loss of connectivity to the SCA or lock structure, DB2 abnormally terminates the members that are affected by the loss. Either abend code 00E30105 or 00F70600 is issued for those members.

For a loss of connectivity to the group buffer pool:

1. DB2 adds entries to the logical page list, if necessary.
2. Applications that are running on the members that lost connectivity continue processing until the next COMMIT point. On the next attempt to access a GBP-dependent page set associated with a disconnected group buffer pool, applications receive a -904 SQLCODE (reason code 00C20204). For inflight units of recovery, the loss of connectivity is detected immediately, and the application receives a -904 SQLCODE (reason code 00C20220).
3. DB2 disconnects from the group buffer pool and issues message DSNB309I with a reason of FAILURE. The connection enters a failed-persistent state.

**System programmer action:** Fix the problem that is causing the loss of connectivity. If the problem is not an obvious one (a disconnected link, for example), call IBM Software Support. After the problem is fixed, restart any failed members as quickly as possible to release retained locks.

Consider the following options:
- If connectivity was lost to the SCA or lock structure, causing the member to fail, restart the failed member on another system that has connectivity to the structure.
  
  If connectivity is lost to a group buffer pool, DB2 continues to run. If you want to start the member on another system, you must stop DB2 before you restart it.
- Manually rebuild the structure on another coupling facility using the z/OS SETXCF command:
  
  ```sql
  SETXCF START,REBUILD,STRM=stringname,LOC=OTHER
  ```
  
  After you rebuild the group buffer pool, there might still be logical page list entries for the page sets. To recover these, enter the command START DATABASE after receiving a DSNB311I or DSNB312I for the page set or partition. Issue the command from a member that is connected to the group buffer pool.
  
  To avoid manual intervention next time, lower the REBUILDPERCENT value in the CFRM policy so that the next time you lose connectivity, DB2 can automatically rebuild.

**Problem: group buffer pool structure failure (no duplexing)**

**Symptom:** The following message appears on the console of the member who will be coordinating damage assessment:

```sql
DSNB314I csect-name DAMAGE ASSESSMENT TO BE TRIGGERED FOR GROUP BUFFER POOL gpname REASON=STRFAIL
```

**System action:** A group buffer pool failure restricts access to the data assigned to that group buffer pool; it does not cause all members of the group to fail.

Applications needing access to group buffer pool data are rejected with a -904 SQL return code. Any of the following reason codes can be returned:

- 00C20204
- 00C20205
- 00C20220

See "Problem: all members have lost connectivity" on page 173 for a description of what DB2 does during automatic recovery.

**System programmer action:** Correct the coupling facility failure. For any page sets that were not automatically recovered by DB2, notify the database administrator to recover the data from the group buffer pool by using the command START DATABASE (dbname) SPACENAM (spacename) to remove the GRECP status.

**Problem: group buffer pool structure failure (duplexing)**

**Symptom:** The following message appears on the console of the member who will be coordinating damage assessment:
DSNB314I csect-name DAMAGE ASSESSMENT TO BE TRIGGERED FOR
GROUP BUFFER POOL gpbnname REASON=STRFAIL

System action: A group buffer pool failure restricts access to the data assigned to that group buffer pool; it does not cause all members of the group to fail. Applications needing access to group buffer pool data are rejected with a -904 SQL return code. Any of the following reason codes can be returned:
   00C20204
   00C20205
   00C20220

See “Problem: all members have lost connectivity” on page 173 for a description of what DB2 does during automatic recovery.

System programmer action: Correct the coupling facility failure. For any page sets that were not automatically recovered by DB2, notify the database administrator to recover the data from the group buffer pool by using the command START DATABASE (dbname) SPACENAM (spacename) to remove the GRECP status.

Problem: lock structure failure

Symptom: Locking requests are suspended until the lock structure is rebuilt. If the lock structure cannot be rebuilt, the following message appears:
   DXR136I irlmx HAS DISCONNECTED FROM THE DATA SHARING GROUP

System action: If the structure cannot be rebuilt, all active members of the group terminate abnormally with a 00E30105 abend code.

System programmer action: See message DXR135E for the root cause of the problem and the corrective procedure.

Problem: SCA structure failure

Symptom: The following message appears:
   DSN7502I -DB1A csect-name SCA STRUCTURE FAILURE, ATTEMPT TO REBUILD IS IN PROGRESS.

DB2 suspends processing until the SCA is rebuilt, using information contained in DB2 memory.

If the SCA cannot be rebuilt, the following message appears:
   DSN7504I -DB1A csect-name SCA STRUCTURE structure-name REBUILD UNSUCCESSFUL.

System action: If the rebuild is unsuccessful, all members of the group terminate abnormally.

System programmer action: Check the termination code for the reason that the rebuild was unsuccessful. Correct the problem and then restart the members of the group.

Problem: allocation failure of the group buffer pool

Symptom: The following message appears:
DSNB301E -DB1A csect-name GROUP BUFFER POOL

gbname CANNOT BE CONNECTED

DB2 REASON CODE = reason1

MVS IXLCONN REASON CODE = xxxx0C08

**System action:** Applications needing access to group buffer pool data are rejected with a -904 SQL return code (reason code 00C20204).

If the group buffer pool cannot be allocated in an alternate coupling facility as specified on the preference list of the CFRM policy, no inter-DB2 RW activity can exist on the table spaces, indexes, or partitions that are assigned to this buffer pool. If the group buffer pool that cannot be allocated is group buffer pool 0, no update activity can exist on the DB2 catalog and directory.

**System programmer action:** Use IFCID 0250 in performance class 20 to determine the reason for the allocation failure. If the trace indicates that the reason for the allocation failure is inadequate storage in the coupling facility, you can:

- Change the CFRM policy to decrease the amount of storage for the group buffer pool, or redefine that group buffer pool to a different coupling facility that has more storage. See "Changing the size of the group buffer pool" on page 270 for more information.
- Have the database administrator reassign some of the table spaces or indexes that are using that group buffer pool to a different group buffer pool.

**Problem: storage shortage in the group buffer pool**

**Symptoms:** The following message appears when the group buffer pool is 75% full:

DSNB319A -DB1A csect-name THERE IS A SHORTAGE OF SPACE IN GROUP BUFFER POOL gbname

If you do not do anything to relieve the shortage, the following message appears when the group buffer pool is 90% full:

DSNB325A -DB1A csect-name THERE IS A CRITICAL SHORTAGE OF SPACE IN GROUP BUFFER POOL gbname

If the group buffer pool is full, DB2 cannot write to it, and the following message appears:

DSNB228I csect-name GROUP BUFFER POOL gbname

cannot be accessed for function

MVS IXLCACHE REASON CODE=xxxx0C17

Performance problems are evidence that the group buffer pool is not large enough. See "Group buffer pool size is too small" on page 263 for more information about such symptoms and how to avoid having writes to the group buffer pool fail because of a lack of storage.

**System action:** DB2 initiates castout processing if it is not already in progress. DB2 then tries again to write to the group buffer pool. For simplexed group buffer pools, or for the primary of a duplexed group buffer pool, pages that cannot be written to the group buffer pool are added to the logical page list and message DSNB250E is issued.

If it is the secondary group buffer pool that is too full, DB2 does not add pages to the logical page list; instead, it takes the structure out of duplexing mode.
**System programmer action:** If you cannot increase the size of the group buffer pool, use the ALTER GROUPBUFFERPOOL command to decrease the castout thresholds. If decreasing the castout threshold negatively impacts performance, this should only be used as a temporary solution.

**Problem: storage shortage in the SCA**

**Symptoms:** The following message appears:

DSN7505I -DB1A csect-name THERE IS A SHORTAGE OF FREE STORAGE IN SCA STRUCTURE sca-structure-name.

If you do not do anything to reclaim space, such as recovering pages from the logical page list, the following message appears when the SCA is 90% full:

DSN7512A -DB1A csect-name THERE IS A CRITICAL SHORTAGE OF FREE STORAGE IN SCA STRUCTURE sca-structure-name.

**System action:** Some critical functions that cannot be completed can cause one or more members of the group to come down with reason code 00F70609.

**System programmer action:** Perform the following steps:

1. Reclaim space in the SCA by removing exception conditions.
   
   You can issue START DATABASE commands with the SPACENAM option, or use the RECOVER utility to remove pages from the logical page list.

2. Restart any failed members.

If these actions do not free up enough space, or if this problem continues to occur, you have the following options, depending on what level of z/OS and coupling facility you have.

- **If both** of the following conditions are true:
  - The SCA is allocated in a coupling facility with a CFLEVEL greater than 0.
  - The currently allocated size of the SCA is less than the maximum structure size as defined by the SIZE parameter of the CFRM policy.

  You can enter the following command to increase the size of the SCA:

  SETXCF START,ALTER,STRNAME=DSNDB0A_SCA,SIZE=newsize

  This example assumes that the group name is DSNDB0A, and that newsize is less than or equal to the maximum size defined in the CFRM policy for the SCA structure.

- **If any** of the following conditions are true:
  - The SCA is allocated in a coupling facility with CFLEVEL=0.
  - The allocated size of the structure is already at the maximum size defined by the SIZE parameter in the CFRM policy.

  Then you must:

  1. Increase the storage for the SCA in the CFRM policy SIZE parameter.
  2. Use the z/OS command SETXCF START,POLICY to start the updated policy.
  3. Use the following z/OS command to rebuild the structure:

     SETXCF START,REBUILD,STRNAME=DSNDB0A_SCA

- **If all** members are down, and you cannot increase the size of the SCA, you must do the following:

  1. Delete the SCA structure by using the following command:

     SETXCF FORCE,STRUCTURE,STRNAME=DSNDB0A_SCA

  2. Increase the size of the SCA in the CFRM policy.
3. **Restart DB2 to rebuild** the SCA using group restart, as described in "Group restart" on page 186.

**Problem: storage shortage in the lock structure**

**Symptom:** A DXR170I message indicates when storage reaches 50, 60, and 70% full. However, because this problem is detected along with other timer driven function, some messages with lower percentages might be missing or skipped if the coupling facility is filling up rapidly. The following message appears at increasing thresholds starting at 80% full:

- `DB1A DXR142I irlmx THE LOCK STRUCTURE structure-name IS zzz% IN USE`

**System action:** DB2 continues processing, but some transactions might obtain a "resource unavailable" code because they are unable to obtain locks.

**System programmer action:** First, make sure that no members are down and holding retained locks. Restarting any failed members can remove the locks retained in the coupling facility lock structure and release that space.

If a failed member is not the problem, you have two courses of action:

- Lower the lock escalation values to get fewer locks. You do this either by lowering the value on the LOCKS PER TABLE(SPACE) of installation panel DSNTIPF or by using the LOCKMAX clause of CREATE TABLESPACE.
- **Increase the size of the lock structure as described in "Changing the size of the lock structure" on page 234.**

**Dealocating structures by force**

In a few exceptional cases, you might need to deallocate a structure by force to get DB2 restarted.

When you forcibly deallocate an SCA or lock structure, it causes a group restart on the next startup of DB2. DB2 can then reconstruct the SCA or lock structure from the logs during group restart.

To deallocate structures, use z/OS SETXCF FORCE commands to delete persistent structures or connections. Each DB2 structure requires a different set of commands.

- For the group buffer pools:
  
  `SETXCF FORCE,CONNECTION,STRNAME=strname,CONNAME=ALL`

- For the SCA:
  
  `SETXCF FORCE,STRUCTURE,STRNAME=strname`

- For the lock structure:
  
  `SETXCF FORCE,CONNECTION,STRNAME=strname,CONNAME=ALL`

  `SETXCF FORCE,STRUCTURE,STRNAME=strname`

**Important:** If your site is running z/OS with APAR OA02620 applied, you cannot delete failed-persistent connections to the lock structure unless you also deallocate the lock structure. Deleting failed-persistent connections without also deallocating the associated structure can result in a loss of coupling facility data. This situation can then cause undetectable losses of data integrity. APAR OA02620 protects your site from data corruption problems that can occur as a result of deleting retained locks. In doing so, the APAR also prevents extended outages that would result from long data recovery operations.
APAR OA02620 makes deleting persistent connections and structures easier. When you forcibly deallocate the lock structure, the operating system deletes failed-persistent connections to the structure for you. Instead of issuing the SETXCF FORCE command twice (once for failed-persistent connections to the lock structure and once for the lock structure itself), you need issue it only one time:

SETXCF FORCE,STRUCTURE,STRNAME=strname

Resolution of Transaction Manager indoubt units of recovery

A global transaction is a unit of work that involves operations on DB2. All of the operations that comprise a global transaction are managed by a transaction manager, such as WebSphere Application Server. When the transaction manager receives transactionally demarcated requests from an application, it translates the requests into a series of transaction control commands, which it directs to the participating resource managers.

Example: An application requests updates to multiple databases. The transaction manager translates these update requests into transaction control commands that are passed to several resource managers. Each resource manager then performs its own set of operations on a database. When the application issues a COMMIT, the transaction manager coordinates the commit of the distributed transaction across all participating resource managers by using the two-phase commit protocol. If any resource manager is unable to complete its portion of the global transaction, the transaction manager directs all participating resource managers to roll back the operations that they performed on behalf of the global transaction.

If a communication failure occurs between the first phase (prepare) and the second phase (commit decision) of a commit, an indoubt transaction is created on the resource manager that experienced the failure. When an indoubt transaction is created, a message like this is displayed on the console of the resource manager:

00- 17.24.36 STC00051 DSNL405I = THREAD
- G91EIE35.GFA7.00F62CC4611.0001=217
- PLACED IN INDOUBT STATE BECAUSE OF
- COMMUNICATION FAILURE WITH COORDINATOR 9.30.30.53.
- INFORMATION Recorder IN TRACE RECORD WITH IFCID=209
- AND IFCID SEQUENCE NUMBER=00000001

After a failure, the transaction manager, such as WebSphere Application Server, is responsible for resolving indoubt transactions and for driving any failure recovery. To perform these functions, the server must be restarted and the recovery process initiated by an operator. You can also manually resolve indoubt transactions with the RECOVER INDOUBT command.

To manually resolve indoubt transactions:

1. Display indoubt threads from the resource manager console:
   
   -DISPLAY THD(*) T(I) DETAIL

   This command produces output like this:

   00- 17.01.03 =dis thd(*) t(i) detail
   - 17.01.04 STC00051 DSNV401I = DISPLAY THREAD REPORT FOLLOWS -
   - 17.01.04 STC00051 DSNV406I = INDOUBT THREADS -
   - COORDINATOR STATUS RESET URID
   - AUTHID
   - 9.30.30.53:4007 INDOUBT 0000111F049A
   - ADMF002
   - V437=WORKSTATION=jaijestsvl, USERID=admfo02, APPLICATION NAME=db2jccmain
   - V440-XID=7C7146CE 00000014 00000021 F6EF6F8B

Chapter 5. Operating with data sharing  181
Note that in this example, only one indoubt thread exists. A transaction is identified by a transaction identifier, called an XID. The first four bytes of the XID (in this case, 7C7146CE) identify the transaction manager. Each XID is associated with a logical unit of work ID (LUWID) at the DB2 server. Note the LUWID that is associated with each transaction, for use in the recovery step.

2. Query the transaction manager to determine whether a commit or abort decision was made for each transaction.

3. Based on the decision recorded by the transaction manager, recover each indoubt thread from the resource manager console by either committing or aborting the transaction. Specify the LUWID from the DISPLAY THREAD command.

```
# -RECOVER INDOUBT ACTION(COMMIT) LUWID(217)
```

or:

```
# -RECOVER INDOUBT ACTION(ABORT) LUWID(217)
```

This command produces output like this:

```
00- 17.30.21 =RECOVER INDOUBT ACTION(COMMIT) LUWID(217)
```

4. Display indoubt threads again from the resource manager console:

```
# -DISPLAY THD(*) T(I) DETAIL
```

This command produces output like this:

```
00- 17.30.37 =dis thd(*) t(i) detail
```

Notice that the transaction now appears as a heuristically committed transaction.

5. If the transaction manager does not recover the indoubt transactions in a timely manner, reset the transactions from the resource manager console to purge the indoubt thread information. Specify the IP address and port from the DISPLAY THREAD command.

---

182  Data Sharing: Planning and Administration
Restarting DB2 after termination

After a failure or a normal shutdown of DB2, you can restart DB2 with the command START DB2. You can also choose to have DB2 automatically restart after a failure by using the automatic restart manager of z/OS. See “Automatic restart of z/OS” on page 36 for more information.

During restart, DB2 resolves inconsistent states. Restart is changed for data sharing because of the following:

- Database exception states, which exist solely on the log in a non-data-sharing environment, are on both the SCA and the log in data sharing.
- Locks that are retained in the event of a failure must be processed.
- If the SCA or the lock structure is lost and cannot be rebuilt on another coupling facility, all members of the group come down. If this unlikely event occurs, then DB2 must perform group restart. Group restart is distinguished from normal restart by the activity of rebuilding the information that was lost from the SCA or lock structure. Group restart does not necessarily mean that all members of the group start up again, but information from all non-starting members must be used to rebuild the lock structure or SCA.

In this section:
- “Normal restart for a data sharing member”
- “Restart light” on page 185
- “Group restart” on page 186
- “Phases of group restart” on page 186
- “Protecting retained locks: failed-persistent connections” on page 190
- “Handling coupling facility connections that hang” on page 191
- “Postponing backout processing” on page 193
- “Restarting a member with conditions” on page 194
- “Deferring recovery during restart” on page 193

Normal restart for a data sharing member

Normal restart for a member of a data sharing group is very much the same as for a non-data-sharing DB2 subsystem. This section includes some additional information about locks, because locks that are held by a failed member can affect the availability of data to the other members of the group that are still running DB2 applications.

Active and retained locks

Active locks: When a member is active, the locks that it holds are called active locks. For transaction locks (L-locks), the normal concurrency mechanisms apply, including suspensions and timeouts when incompatible locks are requested for a
resource. For physical locks (P-locks), DB2 uses a negotiation process to control access. For more information about locking mechanisms, see “Improving concurrency” on page 219.

**Retained locks**: The particular concern for availability is what happens to locks when a member fails. For data sharing, active locks used to control updates to data (modify locks) become retained in the event of a failure. This means that information about those locks is stored in the coupling facility until the locks are released during restart. Retained locks are necessary to protect data in the process of being updated from being accessed by another active member of the group. See “Protecting retained locks: failed-persistent connections” on page 190 for information about the failed-persistent connections associated with retained locks.

DB2 has various types of retained locks, among them are L-locks, page set P-locks, and page P-locks. As long as a retained lock is held by a failed member, another member cannot lock the resource in a mode that is incompatible with the mode of the retained lock on that resource. Incompatible requests from other members are suspended if you specify a non-zero value for the RETAINED LOCK TIMEOUT parameter of installation panel DSNTIPI. Incompatible requests from other members are immediately rejected if you specify a value of 0 for this parameter.

In the case of a page set P-lock, it is conceivable that an entire page set could be unavailable. For example, an X mode page set P-lock is retained if the page set is non-GBP-dependent at the time of the failure. Incompatible lock requests from other members can be processed after the retained locks are removed, which occurs during restart. To keep data available for all members of the group, **restart failed members as soon as possible**, either on the same or another z/OS system.

You can restart a failed member in “light” mode to quickly recover the locks held by that member. Restarting in light mode is primarily intended for a cross-system restart in the event of a failed z/OS system. For more information, see “Restart light” on page 185.

**Retained utility ID locks**: When a member is running a utility, it holds a lock on the utility ID (UID) for that utility. That lock is also retained if the member fails. This means that you cannot restart a utility until the failed member is restarted and the retained lock is converted to an active lock.

**When retained locks are reacquired or purged**

During the restart process, DB2 removes its retained locks in one of two ways:

- DB2 converts the lock to active. This is called reacquiring the lock, and it is what DB2 does for page set P-locks.
- DB2 purges the lock. This is what DB2 does for page P-locks and for L-locks.

The process of reacquiring or purging locks can happen at different times in the restart process, depending on the type of retained lock, as shown in Table 47.

<table>
<thead>
<tr>
<th>Lock type</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page set P-lock</td>
<td>Reacquired when page sets are opened for log apply. This generally happens during forward recovery or before restart completes.</td>
</tr>
<tr>
<td>Page P-lock</td>
<td>Purged at the end of forward recovery.</td>
</tr>
<tr>
<td>L-lock</td>
<td>Purged at the end of restart processing.</td>
</tr>
</tbody>
</table>

Table 47. Restart processing for locks
Table 47. Restart processing for locks (continued)

<table>
<thead>
<tr>
<th>Lock type</th>
<th>Processing</th>
</tr>
</thead>
</table>

Note:
1. The earliest that a page set P-lock can become negotiable is at the end of forward recovery. See "Page set P-Locks" on page 243 for more information about negotiation. If no log apply is needed, it can happen later, such as the end of restart processing.

If a member has a non-zero value for RETAINED LOCK TIMEOUT and it requests a lock that is incompatible with a retained lock, its applications can be suspended as long as the retained locks are held. Its request can go through as soon as the retained lock is purged or becomes negotiable. For example, if an application is suspended because of an incompatible retained page set P-lock, that retained lock most likely becomes active and available for negotiation at the end of forward log recovery.

Restart light

To recover the retained locks held by a member, use the LIGHT(YES) clause of the START DB2 command to restart the member in “light” mode. Restart-light mode is not recommended for a restart in place. It is intended only for a cross-system restart of a system that does not have adequate capacity to sustain the DB2 and IRLM in the event of a failed z/OS system.

Restart-light mode does the following:
- Minimizes the overall storage required to restart the member.
- Removes retained locks as soon as possible, except for the following:
  - Locks that are held by postponed abort units of recovery.
  - IX mode page set P-locks. These locks do not block access by other members; however, they do block drainers, such as utilities.
  - Terminates the member normally after forward and backward recovery is complete. No new work is accepted.

A data sharing group started with the light option is not registered with the automatic resource manager (ARM). Therefore, ARM will not automatically restart a member that has been started with LIGHT(YES).

For information about using restart-light mode in an ARM environment, see "Creating an automatic restart policy" on page 36, which describes special considerations for the ARM policies.

Indoubt units of recovery

When a member is restarted in light mode and indoubt units of recovery (URs) exist, DB2 issues message DSNR052I at the end of restart. The member continues to run in light mode to allow the indoubt URs to be recovered.

While the member is in light mode to resolve the indoubt URs, it allows connections only for the purpose of resolving the indoubt URs. Only connection requests that originate from a connection name that has indoubt URs are allowed, and that connection is not allowed to create new threads. No new DDF connections are allowed when DB2 is in restart-light mode.

Two ways exist to resolve the indoubt URs: automatically through resynchronization processing or manually.
• Automatically: Ensure that the appropriate commit coordinators (IMS or CICS subsystems that have indoubt URs on the restart-light member) are started so that they can resynchronize with the member to resolve the indoubt URs.

• Manually: Use the RECOVER INDOUBT command to manually resolve the indoubt URs.

Use the DISPLAY THREAD command to monitor the progress of indoubt UR resolution.

After the last indoubt UR is resolved, the member terminates. Alternatively, if you want to stop the member before all indoubt URs are resolved, you can issue the STOP DB2 command. DB2 issues message DSNR046I to indicate that incomplete URs still exist.

**DB2 commands and restart-light mode**

The following commands are not allowed when a member is started in light mode:

• DISPLAY DATABASE, START DATABASE, STOP DATABASE
• DISPLAY RLIMIT, START RLIMIT, STOP RLIMIT
• SET SYSPARM

**Group restart**

Group restart requires scanning the logs of each member to rebuild the SCA or retained lock information. It is recommended that you have an alternate coupling facility on which these vital structures can be automatically rebuilt in the event of a coupling facility failure. The automatic rebuild that occurs during a coupling facility failure does not require the log scans that group restart does.

During group restart, all restarting members update the SCA or lock structure from information contained in their logs. If you do not issue a START DB2 command for all members of the group, the started members perform group restart on behalf of the non-starting members by reading their logs.

Although one member can perform restart on behalf of the group, you should restart all of the non-quiesced members together, perhaps by using an automated procedure. This shortens the total restart time. Also, because retained locks are held for non-starting members, it is best to start all members of the group for maximum data availability.

Because all members must synchronize at the end of current status rebuild (CSR) and at the end of forward log recovery, the time taken for group restart done in parallel is determined by the member that has the longest CSR and, if the lock structure is lost, by the member that has the longest forward log recovery.

When the members are synchronized after forward log recovery, backward log recovery proceeds in parallel for the started members.

**Phases of group restart**

The phases of group restart are generally the same as in a non-data-sharing environment, with the addition of function for group restart. The phases of group restart vary based on whether the SCA, lock structure, or both are lost, and whether information is needed from the logs of inactive members. Table 48 on page 187 summarizes the phases, depending on which structure is lost.
Table 48. Summary of group restart phases based on which structure is lost

<table>
<thead>
<tr>
<th>SCA lost</th>
<th>Lock structure lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td>Initialization</td>
</tr>
<tr>
<td>CSR (rebuild SCA)</td>
<td>CSR (reacquire page set P-locks)</td>
</tr>
<tr>
<td>Peer CSR (rebuild SCA)</td>
<td>Peer CSR (rebuild page set P-locks)</td>
</tr>
<tr>
<td>Forward-log recovery (rebuild locks)</td>
<td>Forward-log recovery (rebuild locks) or Peer forward recovery (rebuild locks)</td>
</tr>
<tr>
<td>Backward-log recovery</td>
<td>Backward-log recovery</td>
</tr>
</tbody>
</table>

The message output in the following section shows a group restart controlled by member DB3A on behalf of members DB1A and DB2A.

**DB2 initialization**

The initialization phase verifies BSDSs, logs, and the integrated catalog facility catalog. The RBA of the last log record is retrieved and logging is set to begin at the next control interval (CI) following the last RBA. Also during this phase, DB2 determines if the lock structure or SCA is lost and needs to be recovered.

During initialization, you see messages similar to the following messages:

```
$HASP373 DB3AMSTR STARTED

DSNJ127I 0DB3ADB2 SYSTEM TIMESTAMP FOR BSDS= 95.040 13:03:05.32
DSNJ001I 0DB3ADB2 DSNJW007 CURRENT COPY 1 ACTIVE LOG 753
DATA SET IS DSNAME=DSNC410.THIRD.LOGCOPY1.DS01,
STARTRBA=000000000000,ENDRBA=000000167FFF

DSNJ001I 0DB3ADB2 DSNJW007 CURRENT COPY 2 ACTIVE LOG
DATA SET IS DSNAME=DSNC410.THIRD.LOGCOPY2.DS01,
STARTRBA=000000000000,ENDRBA=000000167FFF
DSNJ099I 0DB3ADB2 LOG RECORDING TO COMMENCE WITH
STARTRBA=000000010000

$HASP373 DB3ADBMM1 STARTED
```

**Current status rebuild**

During current status rebuild (CSR), the following tasks are accomplished:

- The SCA is rebuilt from the log by reading it forward from the last checkpoint.
  
  All restarting members add entries to the indoubt transaction ID (XID) list in the
  SCA from information contained in their logs. If an indoubt XID entry cannot be
  added to the SCA, the member abnormally terminates with reason code
  00F70606.

- DB2 determines all outstanding units of recovery (URs) that were interrupted by
  the previous termination.

- If the lock structure is lost, all partition and page set P-locks are reacquired by
  reading information from the log. These locks are retained locks until the end of
  restart.

When a restarting member has completed its own CSR, it checks and waits for
every other member to finish CSR. If non-starting members exist, peer CSR is
performed.

During CSR, you see messages similar to the following messages. (The phrase in
parentheses is not part of the output.)
DSNR001I 00B3A0B2 RESTART INITIATED
DSNR003I 00B3A0B2 RESTART...PRIOR CHECKPOINT RBA=00000000DC4E
DSNR004I 00B3A0B2 RESTART...UR STATUS COUNTS
IN COMMIT=0, INDOUBT=0, INFLIGHT=0, IN ABORT=0
(End of current status rebuild for member DB3A)

DSNR021I 00B3A0B2 DSNRRGRC DB2 SUBSYSTEM MUST PERFORM
GROUP RESTART FOR PEER MEMBERS

Peer CSR
Peer CSR is skipped unless it is necessary to perform group restart on behalf of non-starting members. Peer CSR is not performed on non-starting members that are normally quiesced.

A restarting member can select an inactive member on which to perform peer initialization and peer CSR:
- If the SCA is lost, the restarting member rebuilds SCA information from the information contained in the nonstarting member’s logs.
- If the lock structure is lost, the restarting member reacquires page set and partition P-locks (as retained locks) for the non-starting member. Those locks are now retained locks.

During peer CSR, you see messages similar to the following messages. (The phrases in parentheses are not part of the output.)
DSNR023I 00B3A0B2 DSNRRGRC GROUP RESTART INITIATED TO
RECOVER THE SCA FOR GROUP MEMBER DB1A
DSNR003I 00B3A0B2 RESTART...PRIOR CHECKPOINT RBA=000000201CC4E
DSNR004I 00B3A0B2 RESTART...UR STATUS COUNTS
IN COMMIT=0, INDOUBT=0, INFLIGHT=0, IN ABORT=0
DSNR024I 00B3A0B2 DSNRRGRC GROUP RESTART COMPLETED TO
RECOVER THE SCA FOR GROUP MEMBER DB1A
(End of peer current status rebuild for member DB1A)

DSNR023I 00B3A0B2 DSNRRGRC GROUP RESTART INITIATED TO
RECOVER THE SCA FOR GROUP MEMBER DB2A
DSNR003I 00B3A0B2 RESTART...PRIOR CHECKPOINT RBA=00000000DC4E
DSNR004I 00B3A0B2 RESTART...UR STATUS COUNTS
IN COMMIT=0, INDOUBT=0, INFLIGHT=0, IN ABORT=0
DSNR024I 00B3A0B2 DSNRRGRC GROUP RESTART COMPLETED TO
RECOVER THE SCA FOR GROUP MEMBER DB2A
(End of peer current status rebuild for DB2A)

DSNR022I 00B3A0B2 DSNRRGRC DB2 SUBSYSTEM HAS
COMPLETED GROUP RESTART FOR PEER MEMBERS
(End of peer processing)

When all members have completed CSR—either by performing it on their own or by having a peer perform it for them—the SCA has been rebuilt, and page set and partition P-locks have been reacquired.

Forward-log recovery
In the forward-log recovery phase, DB2 applies log records and completes any database write operations that were outstanding at the time of the failure. It also rebuilds retained locks during this phase by reading that information from the log. Restart time is longer when lock information needs to be recovered during a normal restart, because DB2 needs to go back to the earliest begin UR for an inflight UR belonging to that subsystem. This is necessary to rebuild the locks that member has obtained during the inflight UR. (A normal restart goes back only as far as the earliest RBA that is needed for database writes or is associated with the begin UR of indoubt units of recovery.)
If a problem exists that prevents an object’s log record from being applied (for example, if the disk version of the data could not be allocated or opened), or if the page set is deferred, DB2 adds the relevant pages and page ranges to the logical page list. Only pages affected by the error are unavailable.

When each restarting member has completed its own forward-log recovery, it checks and waits for all members to finish. If there are non-starting members, peer forward-log recovery is performed.

During forward-log recovery, you see messages similar to the following messages:

- **DSNR005I** 00B3ADB2 RESTART...COUNTS AFTER FORWARD RECOVERY
  IN COMMIT=0, INDOUBT=0
  DSNR021I 00B3ADB2 DSNRGRH DB2 SUBSYSTEM MUST PERFORM
  GROUP RESTART FOR PEER MEMBERS

**Peer forward-log recovery**

Peer forward-log recovery is skipped unless it is necessary to rebuild lock information from information contained in inactive, non-quiesced members’ logs. Peer retained-lock recovery requires that DB2 do a peer initialization, a partial CSR phase to rebuild UR status, and then do the forward-log recovery for the non-started member.

During peer forward-log recovery, you see messages similar to the following messages. (The phrases in parentheses are not part of the output.)

- **DSNR005I** 00B3ADB2 RESTART...COUNTS AFTER FORWARD RECOVERY
  IN COMMIT=0, INDOUBT=0
  DSNR026I 00B3ADB2 DSNRGRH GROUP RESTART COMPLETED TO
  RECOVER RETAINED LOCKS FOR GROUP MEMBER DB1A
  (End of peer forward-log recovery for member DB2A)

- **DSNR005I** 00B3ADB2 RESTART...COUNTS AFTER FORWARD RECOVERY
  IN COMMIT=0, INDOUBT=0
  DSNR026I 00B3ADB2 DSNRGRH GROUP RESTART COMPLETED TO
  RECOVER RETAINED LOCKS FOR GROUP MEMBER DB1A
  (End of partial current status rebuild for DB1A)

- **DSNR005I** 00B3ADB2 RESTART...COUNTS AFTER FORWARD RECOVERY
  IN COMMIT=0, INDOUBT=0
  DSNR026I 00B3ADB2 DSNRGRH GROUP RESTART COMPLETED TO
  RECOVER RETAINED LOCKS FOR GROUP MEMBER DB2A
  (End of peer forward-log recovery for member DB2A)

When all members have completed forward-log recovery—either by performing it on their own or by having a peer perform it for them—the lock structure has been rebuilt.
Backward-log recovery

At this point, all forward-log applies have been performed, and inflight, indoubt, and in-abort transactions are protected by locks. During the backward-log recovery phase, DB2 completes recovery processing by reversing all changes performed for inflight and in-abort units of work.

Any updates that cannot be externalized to the group buffer pool or to disk cause the affected pages to be added to the logical page list.

During backward-log recovery, you see messages similar to the following messages. (The phrase in parentheses is not part of the output.)

```
DSNR006I 0083AD082 RESTART...COUNTS AFTER BACKWARD
RECOVERY
INFLIGHT=0, IN ABORT=0
(End of backward-log recovery for member DB3A)
DSNR002I 0083AD082 RESTART COMPLETED
DSN902I 0083AD082 DSNYASC 'START DB2' NORMAL COMPLETION
```

The backward-log recovery messages for other members do not appear until those members are actually started.

Backward-log recovery can occur in parallel for all the started members. No peer backward-log recovery exists; all members must be started to complete backward-log recovery and to release the locks held by inflight and in-abort transactions.

Protecting retained locks: failed-persistent connections

**Important:** The information in this section is not relevant for sites running z/OS with APAR OA02620 applied. With this APAR, you cannot delete failed-persistent connections to the lock structure unless you also deallocate the lock structure.

Deleting failed-persistent connections without also deallocating the associated structure can result in a loss of coupling facility data. This situation can then cause undetectable losses of data integrity. APAR OA02620 protects your site from data corruption problems that can occur as a result of deleting retained locks. In doing so, the APAR also prevents extended outages that would result from long data recovery operations.

**Use extreme care when deleting failed-persistent connections to the lock structure.**

IRLM and XES use failed-persistent connections to the lock structure to track retained lock information. Retained locks might be lost and data integrity might be exposed by arbitrarily deleting a failed-persistent lock structure connection. You should delete failed-persistent lock structure connections only in the following situations:

- Disaster recovery. All DB2 and IRLM-related failed-persistent connections and structures must be deleted before restarting the data sharing group at the remote site. During the restart, DB2 uses the group restart process to rebuild the retained locks from the logs.
- A Parallel Sysplex-wide outage when the lock structure is forced. Failed-persistent connections can be safely forced when all members are down and the lock structure is also forced. During the restart phase of a disaster recovery process, DB2 uses the group restart process to rebuild the retained locks from the logs.
• After a hard failure occurs, such as a check-stop or abnormal re-IPL of a z/OS image that contains an active member and the DB2 or IRLM member has not been restarted.

Do not delete a member's failed-persistent connection just because that member was normally quiesced.

When a member is shut down while holding retained locks, those retained locks are transferred to another member to hold until the original member is restarted. Therefore, although a normally quiesced member does not hold retained locks for itself, it might hold retained locks for another member that was shut down. The following situations can cause a transfer of retained locks:

• Member failure. Assume that three members, DB1A, DB2A, and DB3A, are running normally. If z/OS incurs a hard failure that takes down DB2A, DB2A’s retained locks are transferred to one of the other members; assume that it was DB1A. If DB1A is subsequently shut down normally, you might assume that no locks are held by DB1A and that you can safely delete DB1A’s failed-persistent connection. In actuality, because DB1A is holding the retained locks from DB2A, deleting DB1A’s failed-persistent connection deletes DB2A’s retained locks and exposes the DB2 data to potential data integrity errors.

• A lock structure rebuild when one or more members are down and holding retained locks. This could be caused by either the z/OS SETXCF command or a coupling facility-related failure.

A coupling facility structure rebuild deletes any failed-persistent connections that existed before the rebuild. The retained locks belonging to failed members are recreated and held during the rebuild by one of the active members until the failed members are restarted.

Handling coupling facility connections that hang

When a member abnormally terminates, its connections to coupling facility structures are put into a FAILING state by cross-system extended services for z/OS (XES). The member remains in this FAILING state until all surviving members of the group have responded to the XES Disconnected/Failed Connection (DiscFailConn) event for each structure. XES sends this event to each surviving member of the group so that the surviving members can take the necessary recovery actions in response to the failed member.

After all surviving members of the group perform the necessary recovery actions and provide DiscFailConn responses to XES for a given coupling facility structure, XES changes the failed member’s connection status for that coupling facility structure from FAILING to FAILED PERSISTENT. The member can reconnect to the coupling facility structure during restart when the member’s status is FAILED PERSISTENT.

When you restart the member immediately following a connection failure, the member can attempt to reconnect to a coupling facility structure while its connection is still in a FAILING state. If this occurs, XES denies the reconnect request with a 0C27 reason code. DB2 responds to this by entering a connection-retry loop until the connection succeeds or until it reaches the maximum retry count.

For the SCA, the maximum retry count is 200 times with a 3-second interval between each attempt. For the group buffer pools, the maximum retry count is 5
times with a 10-second interval between each attempt. You might notice a message similar to the following message, which indicates a failed connection attempt:

```
IXL013I IXLCONN REQUEST FOR STRUCTURE DB2GR0W_SCA FAILED.
JOBNAME: DB2VMSTR ASID: 05E1 CONNECTION NAME: DB2_DB2V
IXLCONN RETURN CODE: 0000000C, REASON CODE: 02010C27
```

The preceding message might be displayed multiple times while DB2 is in a connection-retry loop. This is normal.

In rare cases, one or more of the surviving members of a group encounters difficulties in providing the DiscFailConn response to XES for a given coupling facility structure. When this happens, XES issues a message similar to the following message for each member from which it does not receive a response within two minutes:

```
IXL041I CONNECTOR NAME: DB2_DB2M, JOBNAME: DB2MMSTR, ASID: 0086
HAS NOT RESPONDED TO THE DISCONNECTED/FAILED CONNECTION EVENT FOR
SUBJECT CONNECTION: DB2_DB2V.
DISCONNECT/FAILURE PROCESSING FOR STRUCTURE DB2GR0W_SCA
CANNOT CONTINUE.
DIAG: 0000 0000 00000000
```

In extreme cases, the maximum number of connection retries might be reached. If encountered for the SCA, this situation prevents the failed member from restarting and DB2 issues a message similar to the following message:

```
DSN7506A -DB2 V DSN7LSTK
CONNECTION TO THE SCA STRUCTURE DB2GR0W_SCA FAILED.
 MVS IXLCONN RETURN CODE = 0000000C,
 MVS IXLCONN REASON CODE = 02010C27.
```

### Operator actions for handling coupling facility connections that hang

Perform the following actions to recover from coupling facility structure connections that hang.

1. Save a dump of all DB2 and IRLM members along with SDATA= (COUPLE, XESDATA) so that IBM Software Support can determine what is causing the hung connections. See message II10850 for more information.

2. Attempt a REBUILD of the lock structure. This can sometimes clear the condition that is causing the DiscFailConn response to hang. If the REBUILD of the lock structure works, XES issues a message similar to the following message for each group member as it provides the required DiscFailConn response:

```
IXL043I CONNECTION NAME: DB2_DB2M, JOBNAME: DB2MMSTR, ASID: 0086
HAS PROVIDED THE REQUIRED RESPONSE. THE REQUIRED RESPONSE
FOR THE DISCONNECTED/FAILED CONNECTION EVENT
FOR SUBJECT CONNECTION DB2_DB2V,
STRUCTURE DB2GR0W_SCA IS NO LONGER EXPECTED.
```

If the REBUILD does not work, proceed to step 3.

3. Issue the D XCF,STR,STRNM=<strname>,CONNM=<connname> command for the structure or connector that is in the FAILING state. Alternatively, issue the D XCF,STR,STRNM=<strname>,CONNM=ALL command. Both commands display the status of the structures and connectors that are used by XES.

   If this command identifies the unresponsive members, skip to Step 6. If it does not identify the unresponsive members, proceed to Step 4.

4. Attempt a structure REBUILD for the affected structure, if you have not already done this.
5. If the REBUILD hangs, issue the D XCF,STR,STRNM=<strname> command to identify the unresponsive connector.
   This identifies the members that are unresponsive to the REBUILD. These members are probably the same members that are unresponsive to the DiscFailConn event.
6. Cancel and recycle the unresponsive members. The STOP D command might not work because internal DB2 processes are hung, so cancel IRLM or DB2 MSTR.
   As each member terminates, verify that XES issues message IXL043I to indicate that it no longer expects a DiscFailConn response from that member. When all members that owe responses have been stopped, all connections to the SCA should be ACTIVE or FAILED PERSISTENT.
7. Issue the D XCF,STR,STRNM=<strname>,CONNM=ALL command to verify the status of the connections to SCA.
8. Restart all members with FAILED PERSISTENT connections.
   As each member successfully reconnects to the SCA, XES issues message IXL014I. If a problem still exists, proceed to step 9.
9. Stop and restart the systems on which the unresponsive members are running.
   If restarting the system does not fix the unresponsive members, proceed to step 10.
10. Cancel and recycle all connectors to the coupling facility structure. If a problem still exists, proceed to step 11.
11. Stop and restart all systems.
   Refer to z/OS MVS Recovery and Reconfiguration Guide for more information.

Postponing backout processing

As described in Part 4 (Volume 1) of DB2 Administration Guide you can postpone backout processing for in-abort and inflight units of recovery (URs). The advantage to this is the ability to get DB2 up and accepting new work before handling backout processing.

To summarize, two options exist that let you enable and control postponed backout. Both of these options are on installation panel DSNTIPL:

- LIMIT BACKOUT
  YES or AUTO enables postponed backout. AUTO is the recommended value, because you do not have to remember to issue commands to start backout processing.

- BACKOUT DURATION
  A multiplier that indicates how much to process for backout. This option is valid only when you have specified YES or AUTO for LIMIT BACKOUT.

For more information about these options, see Part 2 of DB2 Installation Guide

Why postponed backout works in a data sharing environment

While a member is restarting, it cannot accept new work. If a restarting member has much backout work to perform, and the work is not postponed, the restart can be time-consuming. In a data sharing environment with spare capacity, you can reroute work to another member of the group. However, if the other system does not have enough capacity, or if you are not able to switch workloads because your configuration is such that there is a strong one-to-one relationship between a member and the workload that runs on that member, postponing backout processing for long-running URs can shorten the outage significantly.
What data is unavailable?
One difference between the non-data-sharing and the data sharing implementation of postponed backout is the degree of data unavailability. In non-data-sharing, DB2 places any page set or partition that has pending backout work into a restrictive status called restart pending, which blocks access to data at the page set or partition level.

In data sharing, no restrictive status is set. Access to data with pending backout work is blocked by transaction locks that persist through restart. The following retained locks persist through restart when postponed backout processing is active:

- Retained transaction locks held on page sets or partitions for which backout work has not been completed
- Retained transaction locks held on tables, pages, rows, or LOBs of those table spaces or partitions

The retained transaction locks on any particular page set or partition are freed when all URs using that page set or partition have completed their backout processing. Until that happens, the page set or partition is placed in an advisory status called advisory restart-pending (AREST).

Advisory restart-pending status
To determine if a member has work pending, use the DISPLAY GROUP command. The following statuses indicate that work is pending for that member:

A I This active member has indoubt URs, URs for which backout work is postponed, or both.
Q I This quiesced member has indoubt URs, URs for which backout work is postponed, or both.

To determine whether the pending work is indoubt URs or postponed backout URs, enter DISPLAY THREAD TYPE(POSTPONED) and DISPLAY THREAD TYPE(INDOUBT).

- To recover indoubt URs, use the RECOVER INDOUBT command.
- To recover postponed abort URs, use RECOVER POSTPONED. (If you specify AUTO for LIMIT BACKOUT, DB2 recovers the postponed URs after restart, and you do not have to enter a command.)

To determine what objects have backout work pending, use the DISPLAY DATABASE command. Objects with backout work pending display a status of AREST. The AREST status is removed when the backout processing is complete for the object. You cannot use the command START DATABASE with ACCESS(FORCE) to remove the advisory status.

Utility access to objects in AREST status: Utilities are not restricted by the AREST status, but any write claims held by postponed abort URs on the objects in AREST status prevent draining utilities from accessing that page set.

Restarting a member with conditions
As described in Part 4 (Volume 1) of DB2 Administration Guide, you might, in unusual circumstances, choose to make inconsistent data available for use without recovering it. This might be the case for certain test groups, for example, where data consistency is not important.
Some installations use conditional restart to bypass a long active UR backout, such as might occur when a long-running batch job fails without having issued interim commits. In data sharing environments, this use of conditional restart is not recommended. It is safer and provides better availability to reroute work to another member or to postpone backout processing rather than suffer the total outage necessary for a conditional restart.

If you do perform conditional restart, you need to stop all members of the group except the one that is conditionally restarting to ensure that applications on those other members do not change the data that is not locked by the restarting member.

This section describes two procedures:

- "Procedure for cold starting a member"
- "Procedure for other conditional restarts"

**Procedure for cold starting a member**

Use this procedure for performing a cold start; that is, when STARTRBA = ENDRBA on the DSNJU003 (change log inventory) utility’s CRESTART statement.

1. Stop all other members of the data sharing group.
2. Cold start the chosen member using ACCESS(MAINT). The cold start deallocates the group buffer pools to which this member was connected.
3. Resolve all data inconsistency problems resulting from the cold start. For more information about how to do this, see Part 4 (Volume 1) of DB2 Administration Guide.
4. After you have resolved the data inconsistencies resulting from the cold start, start all the other members and restart this one without ACCESS(MAINT).

**Procedure for other conditional restarts**

Use this procedure when you are truncating the log but you are not performing a cold start.

1. Stop all other members of the data sharing group.
2. Conditionally restart the chosen member using ACCESS(MAINT).
3. Resolve all data inconsistency problems resulting from the conditional start. For more information about how to do this, see Part 4 (Volume 1) of DB2 Administration Guide.
4. After you have resolved the data inconsistencies resulting from the conditional restart, start all the other members and restart this one without ACCESS(MAINT).

**Deferring recovery during restart**

It is possible to defer recovery for an object during restart. If you use the DEFER installation option, that defers the log apply processing of the object for only the member who specified that option. All the pages that would have been applied to disk or the group buffer pool are instead added to the logical page list. This can affect the rest of the group; any member who needs a page that is on the logical page list will not be able to access that page until the object is restarted.

To make those pages available after DB2 restarts, use the START DATABASE command with the SPACENAM option.

Deferring recovery does not change restart time significantly.
Starting duplexing for a structure

When duplexing starts, a period exists in which activity to the structure is quiesced. For this reason, you should start duplexing during a period of low activity in the system.

You can start duplexing for a structure in one of two ways:

- Activate a new CFRM policy with DUPLEX(ENABLED) for the structure. The structure must have one or more actively-connected DB2 instances that support duplexing, and no connectors that do not support duplexing. If the structure is currently allocated, z/OS can automatically initiate the process to establish duplexing as soon as you activate the policy. If the structure is not currently allocated, the duplexing process can be initiated when the structure is allocated.

- Activate a new CFRM policy with DUPLEX(ALLOWED) for the structure. If the structure is currently allocated, use the following command to start the duplexing rebuild:

  SETXCF START,REBUILD,DUPLEX,STRNAME=strname

  If the structure is not currently allocated, wait until it is allocated before starting the duplexing rebuild.

While duplexing is being established, and for the entire time duplexing is in effect, a display of the structure shows the structure as being in DUPLEXING REBUILD. The rebuild phase is called DUPLEX ESTABLISHED, which means that duplexing is truly active for the structure. See Figure 64 on page 259 for an example.

Stopping duplexing for a structure

To stop duplexing, you must first decide which instance of the structure is to remain as the surviving simplex structure. If you have a choice, use the primary structure as the surviving one. For example, the primary group buffer pool has intact page registration information. If you choose the secondary group buffer pool—which does not have intact page registration information—as the surviving structure, all this information is lost. As a result, all locally cached pages revert to an invalid state and these pages must then be refreshed from the group buffer pool or disk.

If you want to switch temporarily to use one of the duplexed structures as a simplex structure, use the following method:

1. **Optional:** If DUPLEX(ENABLED) is specified for the active CFRM policy, activate a new policy specifying DUPLEX(ALLOWED). For the new DUPLEX value to take effect, all other CFRM policy parameters must remain unchanged.

2. Use the SETXCF STOP,REBUILD command, specifying KEEP=OLD to use the primary structure as a simplex structure, or specifying KEEP=NEW to use the secondary structure as a simplex structure.

   For example, the following command reverts to using the primary structure as the simplex structure:

   SETXCF STOP,RB,DUPLEX,STRNAME=strname,KEEP=OLD

   If you do not plan to reestablish duplexing for the structure in the near future, activate a new CFRM policy specifying DUPLEX(DISABLED) for the structure. Doing so, permanently switches to using one of the duplexed structures as a simplex structure.
If you want to perform maintenance on a coupling facility that contains primary or secondary structure instances, use the SETXCF STOP,RB,DUPLEX command and specify the target coupling facility, as documented in "Shutting down the coupling facility."

Shutting down the coupling facility

Create a plan for those cases in which it is necessary to shut down a coupling facility to apply maintenance or perform some other type of reconfiguration. For the least disruptive shutdown, move all of your structures to another coupling facility before shutting it down. This section provides some guidelines for handling this event. For other structures in the coupling facility, see the appropriate product documentation.

Consider the following guidelines:

1. Prepare for the move:
   - Ensure that you have enough room on the alternate coupling facility for all structures you intend to move.
   - Ensure that the preference list for the group buffer pool, SCA, and lock structures contains the alternate coupling facility information.

2. Move simplexed structures to the new coupling facility by using the following command:
   SETXCF START,REBUILD,CFNAME=newcf,LOC=OTHER
   This command rebuilds all structures that allow rebuild in the alternate coupling facility.

3. Deallocate duplexed structures on the original (target) coupling facility by using the following command:
   SETXCF STOP,REBUILD,DUPLEX,CFNAME=targetcf

   If the CFRM policy specifies DUPLEX(ALLOWED) or DUPLEX(DISABLED), the structure goes into simplex mode.

   If the CFRM policy specifies DUPLEX(ENABLED), z/OS might try to automatically restart duplexing. If you have a third coupling facility specified in the CFRM policy, it is possible to continue duplexing during the outage of the target coupling facility. When an operator command causes the structure to drop from duplex to simplex mode, z/OS avoids automatically reduplexing the structure back into the same coupling facility from which one of the duplexed instances of the structure was just deallocated. Instead, it duplexes the structure in the third coupling facility.

4. Return the coupling facility to service by performing the following steps:
   a. Evacuate the target coupling facility.
      SETXCF START,REBUILD,CFNAME=cfname
   b. Perform the maintenance operation on the target coupling facility and bring it back into the Parallel Sysplex.
   c. Repopulate the target coupling facility with the structures that were in it before the coupling facility was brought down.
      SETXCF START,REBUILD,POPULATECF,CFNAME=cfname
Chapter 6. Performance monitoring and tuning

One of the main objectives of the data sharing function of DB2 UDB for z/OS is to increase processing capacity while using the lower cost zSeries Parallel Sysplex technology. Work load capacity is increased by allowing many DB2 subsystems to access shared DB2 data with full integrity. DB2 data sharing has been designed to address this objective while providing balanced performance for a broad range of SQL applications.

DB2 gives you the power of data sharing while avoiding overhead, whenever possible, for such things as global locking and data caching. However, you can take additional action to reduce the performance cost of data sharing. The purpose of this chapter is to describe briefly how data sharing locking and buffer management work and to offer some guidance about how to improve performance in the group.

This chapter describes the following topics:

- “Setting performance expectations” on page 200
- “Monitoring tools” on page 201
- “Improving the performance of data sharing applications” on page 202
- “Improving the response time for read-only queries” on page 203
- “Improving concurrency” on page 219
- “Tuning group buffer pools” on page 236
- “Access path selection in a data sharing group” on page 272
- “Maintaining in-memory statistics in data sharing” on page 273

Setting performance expectations

With data sharing, as you increase throughput by spreading work across more systems, you can expect to see some processing increases due to the cost of managing and controlling data sharing locking, buffers, and data sets. Remember, however, that those increases are limited to that percentage of your workload that is accessing data with inter-DB2 read/write interest.

This section lists the z/OS address spaces involved in distributed database processing with DB2 UDB for z/OS. (ssnm is a placeholder for the actual DB2 subsystem name.) The purpose of each address space is as follows:

- **ssnmMSTR**
  The system services address space is responsible not only for starting and stopping DB2 UDB for z/OS, but for controlling local access to it. Activities that occur in this address space include commit processing after updates, inserts, and deletes; logging; backout processing; and archiving. With DB2 data sharing, writes to the group buffer pool and the global unlocking that occurs during commit processing both occur under service request blocks (SRBs) in this address space.

- **ssnmDBM1**
  The database services address space is responsible for accessing relational databases controlled by DB2 UDB for z/OS. The input and output to database resources is performed on behalf of SQL application programs in this space.
Activities that occur in this address space include prefetching, space management, and deferred write. With DB2 data sharing, the following activities occur under SRBs in this address space:

- Castouts
- Prefetch interactions with the coupling facility
- P-lock negotiation
- Updates to SYSLGRNX
- Group buffer pool checkpoints

- **ssnmDIST**
  The distributed services address space is responsible for that portion of DB2 UDB for z/OS that provides distributed database capabilities: Distributed Data Facility (DDF).
  When a distributed database request is received, DDF passes the request to ssnmDBM1, so that the required database I/O operations can be performed.

- **ssnmSPAS**
  The DB2 stored procedures address space is responsible for processing stored procedures.

- **IRLM**
  The IRLM address space is responsible for controlling access to database resources.
  Activities that occur in this address space include global lock conflict resolution. When a system is running normally, IRLM’s SRB time is substantially lower than either ssnmDBM1 or ssnmMSTR.

- **Allied address space**
  The allied agent's address space is responsible for handling global lock requests and requests to read from the group buffer pool.

---

### Monitoring tools

This section describes the following:

- "Using resource measurement facility reports"
- "Using DB2 trace" on page 201
- "Using DB2 Performance Expert” on page 201

### Using resource measurement facility reports

The resource measurement facility (RMF™) provides single system and Parallel Sysplex views by reporting on collected resource usage data:

- The Sysplex Summary report provides an integrated view of the entire Parallel Sysplex on one screen.
- The Response Time Distribution report contains details about the distribution of response times on a Parallel Sysplex level and includes the capability of zooming into a single system that indicates problems.
- The Coupling Facility reports include information about storage allocation, structure activity, and subchannel activity which allows you to plan for better resource utilization. See Figure 51 on page 230 and Figure 65 on page 260 for examples of Coupling Facility Activity reports.
- The Shared Device report provides information about how disks and tape resources are shared among the different systems in the Parallel Sysplex.
Using DB2 trace

DB2 writes trace records to help you monitor events in the data sharing group. Many trace classes include information about locking and use of the group buffer pool. For more information about the format of trace records, see Appendix D (Volume 2) of *DB2 Administration Guide*.

Using DB2 Performance Expert

OMEGAMON can monitor the use of shared DB2 resources such as the catalog and directory, group buffer pools, and databases for entire data sharing groups as well as for individual members of the group.

- Reports and traces with *member scope* present a group’s instrumentation data member by member, without merging the data. These reports and traces are similar to those generated in a non-data-sharing environment, where each report or trace is produced for an individual DB2 subsystem.
- Reports and traces with *group scope* merge instrumentation data for individual members and present it for the entire group. The traces show events in chronological order and indicate which member is reporting the event, and the reports show events summarized for the entire group.

The following report sets provide both group-scope and member-scope reporting:
- Accounting
- Audit
- Locking
- Statistics

Group-scope reporting is also available in exception processing and graphics: you can define exception thresholds for groups, and you can create graphs showing performance trends for an entire data sharing group.

With OMEGAMON, you can have processor times reported in service units. This allows times from different central processor complex (CPC) models in the group to be normalized before they are reported.

See [Figure 52 on page 231](#) and [Figure 53 on page 233](#) for examples of OMEGAMON reports.

Improving the performance of data sharing applications

Many of the things you currently do for a single DB2 subsystem to improve response time or reduce processor consumption also hold true in the Parallel Sysplex. For example, most of the recommendations for reducing locking overhead described in “Improving concurrency” on page 219 are the same as for a single system.

However, if the processing speed of the CPU is slower than that on which you are currently running, be sure to plan carefully for applications or DB2 utility processes where processor resource consumption represents the significant part of the elapsed time. Some examples are some batch applications, complex queries, and many DB2 utilities.

This section describes ways to make these resource-intensive applications run at their best in the Parallel Sysplex and includes the following topics:

- “General recommendation” on page 202
- “Migrating batch applications” on page 202

Chapter 6. Performance monitoring and tuning 201
• “Using the resource limit facility (governor)” on page 203

Refer to “Improving the response time for read-only queries” on page 203 for information about using query parallelism in a data sharing group. That section also includes information about governing resources in a data sharing group.

**General recommendation**

Take advantage of data partitioning and design for parallel processing whenever possible. DB2 can use parallel processing effectively only when the data is partitioned. See Part 5 (Volume 2) of [DB2 Administration Guide](https://www.ibm.com) for more information about parallel processing for queries and utilities.

**Migrating batch applications**

When migrating a batch application to a data sharing environment, ensure that you account for any changes in processor speed. Be aware of contention that can occur when there are hot spots in the data as it is updated from multiple members of the group.

**Parallel scheduling**

If a significant portion of the elapsed time of a long-running batch application is due to processor resource consumption and you are currently having scheduling problems, consider running more than one copy of the same program in parallel. You can run each copy on a different key range, typically the partitioning key of the main table.

If your batch application cannot be redesigned to run on separate partitions, consider running batch jobs on the CPC in the group that has the fastest single-central processor speed. This approach is recommended only if the application does not access the coupling facility at a high intensity.

To avoid disk contention, make sure the data partitions are placed on disk devices with separate I/O paths. Consider using the PIECESIZE option of the CREATE or ALTER INDEX SQL statements to give you more control over the data set size of nonpartitioning indexes. See Chapter 5 of [DB2 SQL Reference](https://www.ibm.com) for more information about the PIECESIZE option.

**Designing table spaces for heavy insert and update activity**

For more information about both options that are described here, see “Reducing space map page contention” on page 245.

**The MEMBER CLUSTER option:** If your application does heavy sequential insert processing from multiple members of the group, consider putting the data in a table space that is defined with the MEMBER CLUSTER option. The MEMBER CLUSTER option causes DB2 to insert the data based on available space rather than respecting the clustering index or the first index. For sequential insert applications, specifying MEMBER CLUSTER can reduce P-lock contention and give better insert performance at the cost of possibly higher query times.

**The TRACKMOD option:** When an application is rapidly updating a single table space, contention can exist on the space map pages as DB2 tracks changes. DB2 tracks these changes to help improve the speed of incremental image copies. With TRACKMOD NO, DB2 does not track changes and thus avoids contention on the space map. This option is not recommended if you depend on fast incremental image copies for your backup strategy. If your application does heavy sequential inserts, consider also using the MEMBER CLUSTER option.
Using the resource limit facility (governor)

The resource limit facility (governor) controls the amount of processor time that any dynamic, manipulative SQL statement (SELECT, INSERT, UPDATE, DELETE) can consume in DB2. Different members of a data sharing group can use the same or different resource limit specification tables (RLSTs). Each RLST must have a unique name within the data sharing group. For information about using the resource limit facility to control queries using Sysplex query parallelism, see “Resource limits” on page 217.

Controlling the RLST

The commands that control an RLST (DISPLAY RLIMIT, START RLIMIT, and STOP RLIMIT) affect only the member on which the command is issued. The same holds true if you start an RLST at DB2 startup by using a system parameter.

Dropping objects in the resource limit facility

While an RLST is active in any member, you cannot drop any object associated with the RLST.

Restrictions for STOP and START DATABASE

While an RLST is active in any member, you cannot enter a STOP DATABASE command for a database or table space that contains the RLST, nor can you start the database or table space with ACCESS(UT).

Improving the response time for read-only queries

The response time of a complex SQL query can be constrained by processor resources when it runs on a single central processor. To improve the response time, run a complex query in parallel within a member of a data sharing group, or, with Sysplex query parallelism, use the full power of the data sharing group to process the query on multiple members simultaneously. The information in this section is based on information about query parallelism in Part 5 (Volume 2) of DB2 Administration Guide, but this section emphasizes how you can use the full power of the data sharing group to run complex queries.

Applications that are primarily read-only and processor-intensive benefit from Sysplex query parallelism. If a query qualifies for parallel processing, DB2 determines the optimal degree of parallelism at bind time. DB2 can then distribute parts of the query for processing on different members of the data sharing group at run time.

Terminology: The member that is attached to the application that issued the query is the parallelism coordinator or coordinating member. A member who helps process part of the query is a parallelism assistant or assisting member.

How data is returned to the parallelism coordinator: Data is returned to the parallelism coordinator in one of two ways:

- By work files
  
  If the query requires the use of work files (because the data needs to be sorted, for example), the parallelism coordinator can access the data in the work files of the parallelism assistants. Each assistant writes to its own work file, and the coordinator reads the results from the assistants’ work files.

- By cross-system coupling facility (XCF) links

  For data that does not require the use of work files, XCF is used. XCF traffic can be transported using channel-to-channel connections between the CPCs, or by using the coupling facility.
This section describes the following topics:

- “Planning for Sysplex query parallelism”
- “Enabling Sysplex query parallelism” on page 207
- “Monitoring and tuning parallel queries” on page 211
- “Disabling Sysplex query parallelism” on page 218

Planning for Sysplex query parallelism

This section describes the following tasks:

- “Configuring the data sharing group members”
- “Setting workload management goals”
- “Designing the database” on page 206

Configuring the data sharing group members

Before you can use Sysplex query parallelism, you must configure the members of the data sharing group to use it. In summary:

- The query must originate on a member for which YES is specified for the COORDINATOR parameter of installation panel DSNTIPK.

  The COORDINATOR parameter controls whether the member can send parallel tasks to other members of the data sharing group. If the value of the COORDINATOR parameter is NO at run time, the query runs only on that single member. This is shown in field [C] of the OMEGamon accounting report in Figure 43 on page 213.

- To be considered for the role of parallelism assistants, each of the other members must have YES specified in the ASSISTANT parameter of installation panel DSNTIPK.

  The ASSISTANT parameter controls whether the member can receive parallel tasks from other members of the data sharing group. If this member is the parallelism coordinator for a particular query, the value of its ASSISTANT parameter is not relevant.

- Put work files on shared disks that are accessible by all members that can process parallel queries. Keep work files in a separate buffer pool that is not backed by a group buffer pool. Use the same buffer pool number for all members of the data sharing group. At run time, DB2 assumes that all work files are in the same numbered buffer pool.

- Define group-wide goals for workload management of work that originates on a particular member and of work that is processed by that member on behalf of another.

  See “Setting workload management goals.”

- Individual member buffer pools that can be used for processing queries from other members must have a VPXSEQT buffer pool threshold greater than 0. Because VPXSEQT is a subset of VPSEQT and VPSEQT, these values must also be greater than 0.

  See “Buffer pool threshold for parallelism assistants” on page 208 for more information.

- For optimal use of processor resources, designate only one member on a given CPC as an assistant.

  No benefit exists for splitting a query across multiple members on the same CPC.

Setting workload management goals

You should define how you want z/OS to handle the work of Sysplex query parallelism. In addition to classifying work that runs on the coordinating member, you must also classify work that runs on assisting members. If you do not classify
a member as an assistant, that part of the query that runs on the assistant is discretionary work. Discretionary work runs at the priority usually reserved for very low-priority batch work.

To set workload management goals for parallel queries with which a member assists, you must:

1. Define a service class that is used for queries that run on the assistant.
   This service class can have the same goals as those of the coordinator. Alternatively, you can apply modified goals to the assistants to take into account the numerous amount of enclaves per assistant for a single query. See “Example: setting goals for the parallelism assistants” on page 206 for information about defining a service class.

2. Install the service definition and activate the service policy.

For more information about using workload management, see z/OS MVS Planning: Workload Management.

**How period switches work on parallelism assistants:** Work performed within a single system is governed by periods. Each period has a service unit value and a priority, both of which are assigned by the system. When a piece of work exceeds the service units for a given period, a period switch occurs and the piece of work is switched to the next period, which probably has a lower priority attached to it. The dispatching priority is reevaluated after a period switch.

With Sysplex query parallelism, the work that is associated with a particular query always starts in the first period on each assistant with no accumulated service units.

**Recommendation:** Initially, classify work for the assistants using only one performance period. You can add more performance periods after you monitor the system.

**Example: setting goals for the parallelism coordinator:** Although this work has probably already been done for the parallelism coordinator, this section illustrates the relationship between service classes for the coordinator and for a parallelism assistant. Figure 33 is an example of how you might classify complex queries that originate from TSO. Note that SUBSYSTEM TYPE is TSO.

---

**Figure 33. Defining workload management information for the parallelism coordinator**
The service classes shown in Figure 33 on page 205 are used for queries that originate and run on a particular member.

Example: setting goals for the parallelism assistants: Figure 34 shows one way to classify work on an assistant. In Figure 34, all query work originating from a given environment (such as TSO) has the same goals, but each work environment has its own goals.

![Figure 34. Classifying query work on the assistants.](image)

Figure 34. Classifying query work on the assistants. The DB2 in the SUBSYSTEM TYPE field specifies that these classifications are for query work originating from another member of the data sharing group.

Figure 35 shows another way that work can be classified. It more clearly illustrates the relationship between the work that is classified on the parallelism coordinator in Figure 33 on page 205 and the work that is classified on the parallelism assistant. The SUBSYSTEM TYPE is DB2 and the service class names are different.

![Figure 35. Classifying work for the parallelism assistants](image)

Figure 35. Classifying work for the parallelism assistants

**Designing the database**

Ideally, you want as much partitioning of table spaces as possible and as much separation of I/O as possible. Part 5 (Volume 2) of *DB2 Administration Guide* contains some detailed guidance on how to partition to achieve your performance goals.

**Define table spaces and indexes with GBPCACHE**: It is unlikely that members that are processing a large query will repeatedly read the same pages, so
no need exists to cache those pages in the group buffer pool. Define the relevant table spaces and indexes with GBPCACHE CHANGED (the default).

**Enabling Sysplex query parallelism**

This section describes how to enable parallelism within an application and within the system.

**Enabling parallel processing within an application**

Your application requires no changes to enable parallel processing. Simply use the following settings for parallel processing of static and dynamic queries.

- If the query is static:
  - Use DEGREE(ANY) in a BIND or REBIND subcommand.
  - Use PARALLEL DEGREE ==> ANY in the DB2I DEFAULTS panel for a BIND, REBIND PACKAGE, or REBIND PLAN subcommand.

- If the query is dynamic:
  - Use SET CURRENT DEGREE = 'ANY' in a previous SQL statement.

  **Tip:** Installation panel DSNTIPF allows you to set the default degree (1 or ANY) for the CURRENT DEGREE special register.

**Not all queries are eligible:** Even if you turn on parallelism, not all read-only queries are eligible. For example, queries must be bound with isolation UR or CS. Queries with result sets that contain LOBs are also not eligible for Sysplex query parallelism. To get the effect of RR or RS isolation with Sysplex query parallelism, bind with isolation CS or UR, and use a LOCK TABLE table-name IN SHARE MODE statement before the query.

See Part 5 (Volume 2) of [DB2 Administration Guide](#) for more information about restrictions on queries.

**Enabling parallel processing within a data sharing group**

For a member to be a parallelism coordinator, you must specify YES for the COORDINATOR parameter of installation panel DSNTIPK. To allow a member to be a parallelism assistant, you must specify YES for the ASSISTANT parameter of the same panel.

At run time, assistants must have buffer pools defined to allow parallelism, or DB2 does not send work to those members.

**Displaying coordinator and assistant parameter values:** The command DISPLAY GROUP with DETAIL option allows you to see the values of the COORDINATOR and ASSISTANT subsystem parameters for all active members of a data sharing group. For example, the following command displays output similar to Figure 36 on page 208:

```
-DB1A DISPLAY GROUP DETAIL
```
See DB2 Command Reference for more information about the DISPLAY GROUP command.

**Buffer pool threshold for parallelism assistants**

Use the *assisting parallel sequential threshold* (VPXPSEQT) value to tell DB2 how much of the buffer pool parallel sequential threshold (VPPSEQT) applies to assisting a coordinating member with processing parallel queries. Figure 37 on page 209 shows the relationship among the various buffer pool thresholds.
Sample configurations: The following sample configurations show how the buffer pool parallel allocation threshold values might be set up for two different configurations.

Figure 38 on page 210 shows how the online systems (IMS and CICS) are configured for maximum transaction throughput. These systems send all of their parallel queries to the query processor members on CPC7 and CPC8, and they are not configured to act as parallelism assistants.

- Their COORDINATOR subsystem parameters are set to YES.
- Their ASSISTANT subsystem parameters are set to NO.

The TSO and BATCH systems are configured to run their own queries in parallel, and they can send parallel queries to the query processor members on CPC7 and CPC8. Like the online systems, they are not configured to act as parallelism assistants.

- Their COORDINATOR subsystem parameters are set to YES.
- Their ASSISTANT subsystem parameters are set to NO.

The query processor systems are the only members that have allocated buffer pool space to assisting other members with parallel query processing.

- Their COORDINATOR subsystem parameters are set to YES.
- Their ASSISTANT subsystem parameters are set to YES.
Figure 39 shows a data sharing group in which all members can coordinate and assist with parallel queries. The COORDINATOR and ASSISTANT subsystem parameters on all members are set to YES.

Displaying the buffer pool thresholds: The DB2 command DISPLAY BUFFERPOOL displays all buffer pool thresholds, including the assisting parallel sequential threshold, as shown in Figure 40 on page 211.
Monitoring and tuning parallel queries

This section covers the following topics:
- “Using DISPLAY THREAD”
- “Using a performance monitor” on page 212
- “How DB2 reports what it did” on page 212
- “Monitoring processor use” on page 213
- “Improving response time” on page 213
- “Controlling resources used by parallel operations” on page 216

Using DISPLAY THREAD

DISPLAY THREAD can display information about parallel tasks, and associate them with their originating tasks. If a thread is participating in the processing of a parallel query, issuing a DISPLAY THREAD on any parallelism assistant shows you the thread token of the originating task and the name of the coordinating member. A status of PT indicates a parallel task. All parallel tasks are displayed immediately after their corresponding originating thread.

See Part 3 of *DB2 Command Reference* for information about the syntax of the command DISPLAY THREAD.

Display on the parallelism coordinator: Figure 41 shows an allied, non-distributed originating thread (TOKEN=30) that is established (plan allocated) along with all of its parallel tasks, which are running on members DB1A and DB2A. Because the originating thread is running on DB1A, DB1A is the parallelism coordinator.

Figure 40. Displaying buffer pool thresholds. In this particular buffer pool, 50% of the buffer space is available for parallel processing. All of that parallel space is available to assist with processing queries from other members of the data sharing group.

Figure 41. Display on the parallelism coordinator
Display on the parallelism assistant: Figure 42 shows the display on a parallelism assistant. The PARALLELISM COORDINATOR field tells you which member is the coordinator. The ORIGINATING TOKEN field identifies the originating task on the coordinator.

- 17.10.12 -DB2A DISPLAY THREAD(+)
- 17.10.12 STC00044 DSNV401I -DB2A DISPLAY THREAD REPORT FOLLOWS -
- 17.10.12 STC00044 DSNV402I -DB2A ACTIVE THREADS -
- NAME ST A REQ ID AUTHID PLAN ASID TOKEN
- BATCH PT * 641 PUPPYDML USER001 DSNTEP2 002D 6
- V443-PARALLELISM COORDINATOR=DB1A, ORIGINATING TOKEN=30
- BATCH PT * 72 PUPPYDML USER001 DSNTEP2 002D 7
- V443-PARALLELISM COORDINATOR=DB1A, ORIGINATING TOKEN=30
- BATCH PT * 549 PUPPYDML USER001 DSNTEP2 002D 8
- V443-PARALLELISM COORDINATOR=DB1A, ORIGINATING TOKEN=30
- BATCH PT * 892 PUPPYDML USER001 DSNTEP2 002D 9
- V443-PARALLELISM COORDINATOR=DB1A, ORIGINATING TOKEN=30
- BATCH PT * 549 PUPPYDML USER001 DSNTEP2 002D 10
- V443-PARALLELISM COORDINATOR=DB1A, ORIGINATING TOKEN=30
- DISPLAY ACTIVE REPORT COMPLETE
- 17.10.12 STC00044 DSN9022I -DB2A DSNDVT '-DISPLAY THREAD' NORMAL
- COMPLETION

Figure 42. Display on the parallelism assistant

Using a performance monitor

To see parallel tasks, you can use an online performance monitor such as RMF Monitor III, which displays information about enclaves. Parallel tasks on assistants execute within an enclave. The RMF monitor shows the classification attributes such as the plan name, package name, SQLID, and so on.

How DB2 reports what it did

DB2 reports information about parallelism at two times:

- At bind time (what it plans to do)
- At run time (what it actually did)

This section describes what happens at bind time and at run time, and what can happen to cause the parallel degree to change at run time.

Bind time activity: As with parallelism in general, DB2 tries to determine a parallel degree and a parallel mode (I/O, CP, Sysplex) at bind time. DB2 looks at processor speeds and the number of central processors on each member when it determines the degree of parallelism.

Use DB2 Visual Explain to view the access path that DB2 plans to use for a particular query. An “X” in the PARALLELISM_MODE column indicates that Sysplex query parallelism is chosen for the explainable statement. If only one member of the data sharing group is active at bind time, DB2 can still choose Sysplex query parallelism. This lets any assistants that are active at run time help with processing the query.

Run-time activity: DB2 can use a different parallel degree at run time if any of the following events occur between bind time and run time:

- The number of active members changes, or the processor configuration changes.
  - This event is recorded on the parallelism coordinator at run time in the QXREPOP1 field of the statistics and accounting traces. (See in Figure 43 on page 213)
The amount of buffer space is not sufficient to handle the optimal degree of parallelism. This event is recorded by the parallelism coordinator at run time in the QXREPOP2 field of the statistics or accounting trace. (See Figure 43 in Chapter 6.)

To determine whether work was sent to parallelism assistants, use the accounting trace. (See Figure 43 in Chapter 6 to see how OMEGAMON displays that information.)

To determine the actual degree that was used to process a specific query, use IFCID 0221 in performance trace class 8 (same as with CP parallelism). The QW0221XC field of IFCID 0221 indicates on how many members the parallel group ran. It also indicates when a particular buffer pool is constrained on a particular member. (A parallel group is a set of consecutive operations that executed in parallel and that have the same number of parallel tasks.) IFCIDs 0222 and 0231 include the names of members that participated in processing that parallel group.

When a member does not have enough buffer pool resources to participate in processing a query, the coordinator must skip that member when it distributes work to the group. For more information, see “Is the buffer pool too small?” on page 214.

Monitoring processor use
As with CP parallelism, the accounting trace record fields for processor execution time for the originating task and all of the parallel tasks must be added together to yield the total processor time used by a DB2 thread. In order to perform this same function with Sysplex query parallelism, the accounting trace records from all members involved in the query must be assembled and used. OMEGAMON reports this total time in its accounting report and normalizes the time to the processor size of the originating task. See Figure 43 for an example of a partial accounting report from OMEGAMON.

<table>
<thead>
<tr>
<th>QUERY PARALLEL</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXIMUM MEMBERS</td>
<td>A 2</td>
</tr>
<tr>
<td>MAXIMUM DEGREE</td>
<td>13</td>
</tr>
<tr>
<td>GROUPS EXECUTED</td>
<td>1</td>
</tr>
<tr>
<td>RAN AS PLANNED</td>
<td>0</td>
</tr>
<tr>
<td>RAN REDUCED</td>
<td>B 1</td>
</tr>
<tr>
<td>ONE DB2 COOR=N</td>
<td>C 0</td>
</tr>
<tr>
<td>ONE DB2 ISOLAT</td>
<td>0</td>
</tr>
<tr>
<td>SEQ - CURSOR</td>
<td>0</td>
</tr>
<tr>
<td>SEQ - NO ESA</td>
<td>0</td>
</tr>
<tr>
<td>SEQ - NO BUF</td>
<td>D 0</td>
</tr>
<tr>
<td>SEQ - ENCL.SER</td>
<td>0</td>
</tr>
<tr>
<td>MEMB SKIPPED(%)</td>
<td>E 0</td>
</tr>
<tr>
<td>DISABLED BY RLF</td>
<td>NO</td>
</tr>
<tr>
<td>REFORM PAR-CONF</td>
<td>F 0</td>
</tr>
<tr>
<td>REFORM PAR-BUF</td>
<td>G 0</td>
</tr>
</tbody>
</table>

Figure 43. Partial accounting trace that shows Sysplex query parallelism

Improving response time
When DB2 runs a query on more than one DB2 member, the query elapsed time should improve when compared to running the query within a single member. The actual elapsed time improvement is affected by dynamic factors such as processor utilization, DB2 buffer pool availability, I/O contention, and XCF capacity.
As with CP parallelism, to determine whether performance tuning is needed to further improve the elapsed time of a query, examine the elapsed time and CP execution time of each parallel task (via the IFCID 0231 class 8 performance trace record), especially the ones with the largest times.

The following sections describe some questions to ask when you are determining how to improve query response times:

- **“Is the buffer pool too small?”** on page 215
- **“Is there I/O contention?”** on page 215
- **“Is there lock contention?”** on page 215
- **“Is there XCF signaling contention?”** on page 216
- **“Is there inter-DB2 read/write interest?”** on page 216

**Is the buffer pool too small?:** An indicator of buffer pool shortages are non-zero values in the QXREDFGRP (see [B](#)) in Figure 43 on page 213 and QXDEGBUF (see [D](#)) in Figure 43 on page 213 fields in the statistics or accounting trace record. If response time goals are not being met, further analysis can help you determine which buffer pools are causing the problems. Use performance class 8 and inspect the IFCID 0221 trace record to pinpoint which buffer pools are too small. You might need to increase the size of the buffer pools or increase the number of buffers available to assist with processing query work.

For more information about determining buffer pool size for parallel work, see Part 5 (Volume 2) of [DB2 Administration Guide](#).

**Reformulating the degree:** When buffer pool resources are scarce, DB2 reformulates the degree of parallelism (indicated by [G](#)) in Figure 43 on page 213 and determines a distribution scheme that works best with the reformulated degree. [G](#) is incremented once for each parallel group that was reformulated.

The percentage of members that were unable to perform some or all of the work is indicated by [E](#) in Figure 43 on page 213.

The interaction of these two fields ([G](#) and [E](#)) is best illustrated by an example. Assume that you have a four-member data sharing group of similar processor models with a planned degree of 40. In this case, DB2 might send 10 parallel tasks to each member. However, if buffer pool resources are scarce on two of the members and those members can only process five parallel tasks each, DB2 can reduce the parallel degree to 30, incrementing [G](#) once and setting [E](#) to 50%.

**Work file buffer pools:** Do not forget to set the parallel sequential threshold for work file buffer pools. DB2 assumes that all of your work files are of the same size (4 KB or 32 KB) in the same buffer pool number on all members, and it makes run time decisions based on a single buffer pool. A lack of buffer pool resources for the work files can lead to a reduced degree of parallelism or cause the query to run sequentially.

**Is there I/O contention?:** One possible cause of poor response time is contention for I/O resources. Contention can occur in any place in the I/O subsystem. Here are some ways to determine if and where I/O contention is causing the problem:

1. **Monitor**
   If you have previous accounting reports, look for changes in those reports. If the application has not changed (that is, the SQL Profile and the number of GETPAGEs per commit are relatively constant), refer to the next step.

2. **Analyze**
If you see increased class 3 I/O times from the accounting reports (specifically the SYNCHRON I/O and OTHER READ I/O fields from a OMEGAMON report), check whether the number of I/O operations per commit has increased. If so, consider some form of buffer pool tuning. By increasing the size of the pool, isolating the data into a separate pool, or tuning thresholds, you might be able to reduce the number of I/O operations and speed up the remaining I/O operations in the system.

If tuning the buffer pool does not solve the problem, try to pinpoint the I/O trouble spot. Use the following reports:

- SMF type 42 records, subtypes 5 and 6, show the response times by data set for an interval.
- The RMF Direct Access Device Activity Report shows response time by volume for an interval. It also details where in the I/O subsystem the time is spent. The detail report is of the components of the average response time to the volume. Use this information to find the bottlenecks in the setup or capacity of the I/O subsystem.

3. Relieve the constraint

When you have determined where the problem is occurring, you can take steps to relieve the constraint. This could be a matter of adding to the I/O subsystem by taking such actions as adding extra channel paths to the disk controller, adding storage directors in the disk controller, adding extra cache/NVS in the controller, or adding extra disk volumes. More typically, it means that you have to move data sets from one volume to another.

**Where to place data sets:** In general, you want frequently used data sets or partitions to be allocated across your available disks so that I/O operations are distributed as evenly as possible. Ensure that device and control unit utilization is balanced. This helps balance I/O operations across the I/O configuration and takes maximum advantage of parallel I/O operations.

To determine whether the partitioning of a table or the physical placement of the partitions are reasonable, see performance trace records, IFCIDs 0221 and 0222.

The IFCID 0221 record describes how the tables within a parallel group are partitioned by specifying the key range or page range for each partition. The IFCID 0222 record shows the elapsed time statistics for each parallel operation. The elapsed time for each parallel operation within a parallel group should be comparable. An operation with a much higher elapsed time means that DB2 is performing more I/O operations on a particular logical partition than is desirable, or a significant contention exists on the disk volume or channel path.

If an uneven distribution of work exists and is causing the I/O problems, consider moving data that has low activity close to data that is more frequently accessed. Also be sure that your high-priority work is not sharing I/O resources with work that ties up the I/O subsystem.

If the elapsed times of the parallel tasks are comparable but are still too high, consider repartitioning the table space to have more partitions, assuming that processor time is not a bottleneck.

**Is there lock contention?:** To avoid lock contention, run with an isolation level of UR, if that is possible. If it is not possible to run with an isolation level of UR, try to take advantage of lock avoidance by running with ISOLATION(CS) and specifying CURRENTDATA(NO). This can minimize the locking effect.
Is there XCF signaling contention?: The resource measurement facility (RMF) of z/OS provides an XCF Activity Report that contains useful measurement data for analyzing the performance of the XCF signaling service and for doing capacity planning for a z/OS system in a Parallel Sysplex. The report shows the data collected on a system during Parallel Sysplex processing. Each system collects its own data, and the RMF on each system produces reports only about that system’s data. You might need to run the RMF reports on two or more systems to get data for corresponding outbound and inbound signalling paths in order to better understand the message traffic patterns.

RMF also provides a Coupling Facility Activity report. It provides information about the usage of a coupling facility, structures within a coupling facility, and subchannel connections to a coupling facility in a Parallel Sysplex.

IRLM traces XCF messages in the IRLM Exception (EXP) trace, which you can use to tune XCF performance.

For more information about monitoring XCF activity, see z/OS MVS Setting Up a Sysplex.

Is there inter-DB2 read/write interest?: Because DB2 can split query processing onto different members, updates of the same page set cause inter-DB2 read/write interest as part of the normal data sharing integrity process. To ensure that updates on a page set are seen by the query, DB2 flushes all changed pages from the buffer pool before processing the query on other members.

Inter-DB2 read/write interest can also cause additional locking overhead. Child locks (page or row) are propagated to XES and the coupling facility based on the inter-DB2 interest of the page set P-lock. If the held state of the page set P-lock is IX, indicating inter-system read/write interest, all child locks must be propagated. To avoid locking overheads, use isolation level UR, or try to limit the table space locks to IS on all data sharing members to avoid child lock propagation.

For partitioned table spaces, DB2 can avoid locking all partitions in a table space, thus reducing the number of child locks that must be propagated. To take advantage of this, you must bind the plan or package with ACQUIRE(USE).

Controlling resources used by parallel operations
This section describes several different ways to control the amount of system resources that are used for processing queries using Sysplex query parallelism.

Priority: You can use z/OS workload management to control the priority of parallel query work within a member. Pay special attention to the task of classifying work that runs on parallelism assistants. See “Setting workload management goals” on page 204 for more information about how to do this.

Buffer pool space: You can control how much of the total buffer pool is used for parallel processing in general, and for Sysplex query parallelism specifically.

• The parallel threshold (VPPSEQT) determines the amount of space that is used for all parallel processing.

• The assisting parallel sequential threshold (VPXPSEQT) determines what subset of the parallel space is used to assist queries originating from another member of the data sharing group.

Response time can degrade if these thresholds are set too low.
**Resource limits:** You can use the DB2 resource limit facility (governor) to help control dynamic queries that use Sysplex query parallelism. Resource limits are local in scope, so DB2 can ensure that a specific dynamic query does not overrun a member. Members of a data sharing group can share the same resource limit specification table (RLST), or each member can have its own RLST. In either case, DB2 honors the limits that are specified on that member, no matter how many tasks are included in the statement.

Figure 44 illustrates how dynamic queries that use Sysplex query parallelism are governed by the DB2 resource limit facility.

Statement X: Limit=200 000 service units

<table>
<thead>
<tr>
<th>Parallelism Coordinator</th>
<th>Assistant 1</th>
<th>Assistant 2</th>
<th>Assistant 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 000</td>
<td>200 000</td>
<td>200 000</td>
<td>200 000</td>
</tr>
</tbody>
</table>

Figure 44. Governing a dynamic query that runs on four members. This figure assumes that all members share the same RLST. Statement X is not allowed to consume more than 200 000 service units on any member.

**Governing statements with more than one parallel group:** If multiple parallel groups process queries on a member, each parallel group can consume up to the service unit limit that is set for that member. In contrast, on the coordinator, all parallel groups, including the originating task, are governed by the limit on that coordinating member. Figure 45 illustrates how a query with two parallel groups runs at the same time.

Statement X: Limit=200 000 service units
Statement Y: Limit=200 000 service units

<table>
<thead>
<tr>
<th>Parallelism Coordinator</th>
<th>Assistant 1</th>
<th>Assistant 2</th>
<th>Assistant 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 000</td>
<td>200 000</td>
<td>200 000</td>
<td>200 000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parallel task (X) in parallel group 1</th>
<th>Parallel task (Y) in parallel group 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 000</td>
<td>200 000</td>
</tr>
</tbody>
</table>

Figure 45. A query with two parallel groups running at the same time. This figure assumes that members are sharing the same RLST. No parallel group that is participating in the processing of statement Y can consume more than 200 000 service units. On the parallelism coordinator, all tasks in all parallel groups are constrained by the limit specified for the statement.

**Queries that use INSTALL SYSADM or SYSOPR authorities:** If a query is submitted by an authorization ID with INSTALL authority, none of the parallel tasks is governed, regardless of where the parallel tasks run.

**A statement that is executed more than once:** If the same statement executes more than once in an application, the service unit limit is applied differently to the parallelism coordinator and assistants. The service unit limit applies to all executions of the statement on the parallelism coordinator such that the cumulative
number of service units consumed cannot exceed the limit. Conversely, on the
parallelism assistants, each execution of the statement is subject to the service unit
limit. The number of service units consumed each time the statement executes
cannot exceed the limit.

**Disabling Sysplex query parallelism**

In addition to controls that enable or disable parallelism, you can control Sysplex
query parallelism specifically. You can disable Sysplex query parallelism on a
system-wide basis by:

- Specifying COORDINATOR=NO and ASSISTANT=NO on installation panel
  DSNTIPK.
- Using buffer pool threshold controls that are described in "Buffer pool threshold
  for parallelism assistants" on page 208.

You can also disable Sysplex query parallelism for a single dynamic query by
specifying a value of '5' for the resource limit facility (governor).

**Table 49** summarizes the controls for disabling Sysplex query parallelism.

**Table 49. Controls for disabling Sysplex query parallelism**

<table>
<thead>
<tr>
<th>Control</th>
<th>Checked at...</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>COORDINATOR subsystem parameter</td>
<td>Bind and run time</td>
<td>Set the value of COORDINATOR to NO to restrict parallelism to this single member. (See the QXCOORNO field of IFCID 0002 for cases in which the parameter was changed between bind and run time.) Changing the value of this parameter from NO to YES requires that plans and packages be rebound before they are considered for Sysplex query parallelism.</td>
</tr>
<tr>
<td>ASSISTANT subsystem parameter</td>
<td>Bind and run time</td>
<td>Set the value of ASSISTANT to NO to prevent this member from being considered as a parallelism assistant. (At run time, the assistant's buffer pool must be defined to allow parallelism; otherwise, the coordinator does not send work there.) Changes from NO to YES require that plans or packages be rebound for this assistant's processing capability to affect the planned parallel degree.</td>
</tr>
<tr>
<td>Parallel buffer pool threshold (VPPSEQT) of the coordinator</td>
<td>Run time</td>
<td>Set this value to 0 to disallow query processing on this member.</td>
</tr>
<tr>
<td>Assisting parallel sequential threshold (VPXPSEQT)</td>
<td>Run time</td>
<td>Set this value to 0 to prevent this member from being considered as a parallelism assistant for any query that is using that buffer pool. To disallow the entire subsystem from assisting with Sysplex query parallelism, all buffer pools must have VPXPSEQT=0 or VPPSEQT=0.</td>
</tr>
</tbody>
</table>
Table 49. Controls for disabling Sysplex query parallelism (continued)

<table>
<thead>
<tr>
<th>Control</th>
<th>Checked at...</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Governor (RLFFUNC=’5’)</td>
<td>Run time</td>
<td>Set this value to 5 to disable Sysplex query parallelism for a single dynamic query.</td>
</tr>
<tr>
<td>BIND option DEGREE (1)</td>
<td>Bind time</td>
<td>Specify 1 for this BIND option to disable query parallelism for static queries.</td>
</tr>
<tr>
<td>Special register SET CURRENT DEGREE=’1’ or a 1 in the CURRENT DEGREE parameter of installation panel DSNTIPF</td>
<td>Run time</td>
<td>Set this value to 1 to disable query parallelism for dynamic queries.</td>
</tr>
</tbody>
</table>

Improving concurrency

Transaction locks are often called logical locks (L-locks) in data sharing. This section briefly describes how transaction locking works in a data sharing environment, and highlights some actions that you can take to reduce locking overhead. It includes the following topics:

- “Global transaction locking”
- “Tuning your use of locks” on page 222
- “Tuning deadlock and timeout processing” on page 226
- “Monitoring DB2 locking” on page 229
- “Changing the size of the lock structure” on page 234

Data sharing also makes use of physical locks (P-locks). But, P-locks are related to caching, not concurrency, and they use different mechanisms than the transaction locks you are familiar with in DB2. See “Using P-locks” on page 243 for information about P-locks.

Global transaction locking

With data sharing, concurrency control exists both within a specific member and among all members of a data sharing group. This means that locks used in data sharing are global in scope. Many global locks are processed not only by the local IRLM but also by the cross-system extended services (XES) of z/OS and by the lock structure in the coupling facility.

Locking optimizations

DB2 has the following optimizations, which reduce the necessity of processing locks beyond the local IRLM whenever possible:

- Explicit hierarchical locking, described in “Explicit hierarchical locking” on page 220, makes certain optimizations possible. When no inter-DB2 read/write interest exists in an object, it is possible to avoid processing certain locks beyond the local IRLM.
- If a single member with update interest and multiple members with read-only interest exist, DB2 propagates fewer locks than when all members have update interest in the same page set.
- All locks (except L-locks) that are held beyond the local IRLM are owned by a member, not by an individual work unit. This reduces lock propagation by requiring that only the most restrictive lock mode for an object on a given member be propagated to XES and the coupling facility. A new lock that is equal
to, or less restrictive than, the lock currently being held is not propagated. "A locking scenario" on page 221 provides an example of this type of optimization.

- When the lock structure is allocated in a coupling facility of CFLEVEL=2 or higher, IRLM can release many locks with just one request to XES. This can occur, for example, after a transaction commits and has two or more locks that need to be unlocked in XES. It also can occur at DB2 shutdown or abnormal termination when the member has two or more locks that need to be unlocked.

**Explicit hierarchical locking**

When sharing data, DB2 uses explicit hierarchical locking to determine whether propagating L-locks beyond the local IRLM to XES and to the coupling facility is necessary. Explicit hierarchical locking allows IRLM to grant child locks locally when no inter-DB2 read/write interest exists on the parent. Granting lock requests locally, versus globally, improves performance.

The top object in the hierarchy is a parent; all objects below the parent are children, with the caveat that a child can be the parent of another child. While a lock is held, the first lock on the top parent is always propagated to XES and the lock structure. Thereafter, only more restrictive locks are propagated. When the lock is released, the process begins again. For partitioned table spaces, each locked partition is a parent of the child locks that are held for that partition. Explicit hierarchical locking is based on the lock hierarchy that is shown in Table 50.

**Table 50. Lock hierarchy.** Indexes are not included in the hierarchy because index pages are protected by locks on the corresponding data.

<table>
<thead>
<tr>
<th>Parent</th>
<th>Children or child</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple table space</td>
<td>Data pages and rows</td>
</tr>
<tr>
<td>Partitioned table space</td>
<td>Data pages and rows</td>
</tr>
<tr>
<td>Segmented table space</td>
<td>Tables, data pages, and rows</td>
</tr>
<tr>
<td>LOB table space</td>
<td>LOB</td>
</tr>
</tbody>
</table>

Locks on child objects are propagated depending on the compatibility of the page set P-lock with the page set P-locks that are held by other members for the table space.

**Table 51.** shows the conditions that cause the child locks to be propagated. "A locking scenario" on page 221 describes explicit hierarchical locking in action.

**Table 51. Determining when child locks are propagated to XES**

<table>
<thead>
<tr>
<th>Maximum lock mode of the member is ...</th>
<th>And the maximum lock mode of other members is ...</th>
<th>Are X children propagated?</th>
<th>Are S children propagated?</th>
<th>Are U children propagated?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS, S</td>
<td>None, IS, S</td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>None</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>IS</td>
<td>IX, SIX</td>
<td>N/A</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>IX, SIX</td>
<td>IS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IX</td>
<td>IX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IX, SIX</td>
<td>None</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 51. Determining when child locks are propagated to XES (continued)

<table>
<thead>
<tr>
<th>Maximum lock mode of the member is</th>
<th>And the maximum lock mode of other members is...</th>
<th>Are X children propagated?</th>
<th>Are S children propagated?</th>
<th>Are U children propagated?</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-locks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-locks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U-locks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:
- Some child L-locks might be acquired before the page set P-lock is obtained. When this happens, child L-locks are automatically propagated.
- When a page set switches from inter-system read/write interest to no inter-system read/write interest, a short period of time exists when the page set remains GBP-dependent, before the P-lock reverts to X state. During this time, child L-locks continue to be propagated.

Relationship between L-locks and P-locks: L-locks and P-locks are managed independently of one another. Inter-DB2 interest is possible on an L-lock but not on a P-lock that is held on the same object. Inter-DB2 interest is also possible on the P-lock but not on the L-lock. The inter-DB2 interest on the page set P-lock controls both GBP-dependency and child lock propagation.

A locking scenario

Figure 46 shows an example of locking activity between two members of a data sharing group.

![Figure 46. A lock propagation scenario](image_url)

Relationship between L-locks and P-locks: L-locks and P-locks are managed independently of one another. Inter-DB2 interest is possible on an L-lock but not on a P-lock that is held on the same object. Inter-DB2 interest is also possible on the P-lock but not on the L-lock. The inter-DB2 interest on the page set P-lock controls both GBP-dependency and child lock propagation.

A locking scenario

Figure 46 shows an example of locking activity between two members of a data sharing group.

In Figure 46:
1. The L-lock on table space 1 (TS1), which is associated with transaction 2, is not propagated because DB2A already holds a lock of an equal restriction on that object. (The L-lock on TS1, which is associated with transaction 3, is propagated because that lock is from another member.)
2. The child L-lock on page 1, which is associated with transaction 1, is not propagated at the time that it is requested because its parent lock is IS on both members: no inter-DB2 read/write interest exists on the parent.
3. When Transaction 4 upgrades its lock to IX on TS1, its X lock on Page 2 must be propagated because inter-DB2 read/write interest now exists on the parent. Also, the child lock on Page 1, which is associated with transaction 1, must be propagated.
The child lock is propagated under an IRLM SRB, not under the transaction’s TCB. This propagation is counted in the statistics report as an asynchronous propagation, as shown in [8] in Figure 52 on page 231.

**Determining whether locks have been propagated**

The statistics and accounting traces indicate the number of global locks that have been propagated to XES. The ratio of this number to the total number of global locks requested reflects the effects of explicit hierarchical locking and other locking optimizations. See “Measuring transaction locking optimizations” on page 233 for more information.

**Tuning your use of locks**

Most recommendations for reducing lock contention and locking costs in a single system hold true when sharing data. This section reiterates some general recommendations and emphasizes the following topics:

- “Avoiding false contention” on page 223
- “Reducing the time needed to resolve contentions” on page 225
- “Avoiding partition locks on all table space partitions” on page 225

**General recommendations**

To reduce locking contention, use the same tuning actions that are in place for single-DB2 processing, as described in Part 5 (Volume 2) of DB2 Administration Guide. Here is a summary:

- Use partitioned table spaces.
- Use page locking.

Although row locking can increase concurrency in some cases, you must weigh the benefits of increased concurrency against the increase in locking overhead that row locking might incur. (The amount of overhead depends on how well your application can avoid taking locks.)

One way to achieve the concurrency of row locking and avoid the additional data sharing lock overhead is to define the table space with MAXROWS 1. See DB2 SQL Reference for more information.

- Use an ISOLATION level of uncommitted read (UR), if your applications can tolerate reading uncommitted data.
- Take advantage of lock avoidance whenever possible, if you cannot use ISOLATION(UR). You do this by binding with an isolation level of cursor stability (CS) and CURRENTDATA(NO). These are not the defaults.
- Reduce the scope of BIND operations by using packages. This reduces DB2 catalog and directory contention.
- Design for thread reuse and choose the RELEASE option carefully.

If you use the BIND option RELEASE(DEALLOCATE) for objects that do not have a lot of concurrent activity within a member, you can avoid the cost of releasing and reacquiring the same parent locks again and again. You can also reduce the amount of false contention, described in “Avoiding false contention” on page 223 for those transactions that use the thread.

To achieve a good balance between storage and processor usage, use the bind option RELEASE(DEALLOCATE) for plans or packages that are frequently used. To avoid increasing the EDM pool storage too much, use RELEASE(COMMIT) for plans or packages that are not used as frequently.

- Design for selective partition locking, and bind with ACQUIRE(USE). For more information about selective partition locking, see Part 5 (Volume 2) of DB2 Administration Guide.
Avoiding false contention

The coupling facility lock structure has two parts: a lock table, which is used to determine whether inter-DB2 read/write interest exists on a particular resource, and a list of the update locks that are currently held. When considering false contention, you must be concerned with the size of the lock table. The total size of the lock structure determines the size of the lock table. Assuming that you specify an INITSIZE value on the CFRM policy that is a power of 2, the lock table is allocated to one-half the total size of the lock structure. The value that you specify for the lock table entries (LTE) parameter in the IRLMPROC or with the MODIFY irlmproc,SET,LTE command can control how the lock structure is partitioned. The number of members in the group determines the size of each entry in that lock table. IRLM uses the value that you specify in the LOCK ENTRY SIZE parameter of installation panel DSNTIPJ to determine the initial size of the lock table entries. IRLM also uses the value that you specify in the NUMBER OF LOCK ENTRIES parameter of installation panel DSNTIPJ to determine how the lock structure is initially partitioned. See Part 2 of [DB2 Installation Guide](#) for details.

IRLM assigns locked resources to an entry value in the lock table. This is how it can quickly check to see if a resource is already locked. If the lock structure is too small (thereby making the lock table portion too small), many locks can be represented by a single value. Thus, “false” lock contention can exist. False lock contention is where two different locks on different resources hash to the same lock entry. The second lock requester is suspended until it is determined that no real lock contention exists on the resource.

False contention can be a problem for work loads that have heavy inter-DB2 read/write interest.

Monitoring for false contention: You can determine the amount of false contention by using the RMF Coupling Facility Activity reports as described in “Using the coupling facility structure activity report of RMF” on page 230. DB2 also provides necessary information in its accounting and statistics trace classes. See “Using the DB2 statistics trace” on page 231 and “Using the DB2 accounting trace” on page 232 for more information. More detailed information can be found in the performance trace, as described in “Using the DB2 performance trace” on page 233.

How much contention is acceptable: For the best performance, you want to achieve the least possible amount of global lock contention, both real and false. Aim for total global lock contention of less than 5%, preferably less than 1%. The descriptions of the various reports show you how to determine the contention percentages.

How to reduce false contention: The following tips can help you reduce false contention:

- As much as possible, reduce the amount of real lock contention in your applications.
- Specify a larger size for the lock structure and manually rebuild it as described in “Changing the size of the lock structure by rebuilding” on page 235.
- Ensure that the value for LOCK ENTRY SIZE is not too large for the number of members in your group.

The LOCK ENTRY SIZE parameter for the first IRLM to join the group determines the size of each lock entry in the lock table.
A lock entry size of **two allows** twice as many lock entries as a lock entry size of four, as illustrated in Figure 47.

**Figure 47. Initial lock entry size**

IRLM automatically rebuilds the structure when it needs to. For example, if MAXUSRS=7, when the seventh member joins the group, IRLM automatically rebuilds the lock structure to create the 4-byte lock entries. This prepares the lock structure to handle an eighth member joining the group.

For this reason, even if you anticipate your group growing beyond seven members, you can start out with a lock entry size of two to make the most efficient use of lock structure storage.

**Recommendation:** Set MAXUSRS for all IRLM instances to an appropriate value for the number of members that you will have in a data sharing group. For example, if you plan to have between seven and 22 members in your data sharing group on a regular basis, you should set MAXUSRS=23 in the IRLMPROC procedure for the IRLM instance that is associated with each member of the data sharing group. Alternatively, you can set the LOCK ENTRY SIZE field value to 4 in installation panel DSNTIPJ when you install each member of the data sharing group. Doing this causes the DB2 installation process to set the MAXUSRS value to 23 in the associated IRLMPROC procedures. Lock structure allocation occurs when the first member joins the data sharing group after an IPL of the operating system, or after any situation that causes the lock structure to be deallocated.

If you initially set MAXUSRS=23 for all IRLM instances, the lock structure size is already adequate for up to 22 members, no matter which IRLM is the first to connect. A lock structure rebuild is not necessary when the seventh member joins the group.

If you increase the lock entry size (and thereby increase MAXUSRS), you should increase the lock structure size, to maintain the number of lock table entries and record list entries. If you do not increase the lock structure size, IRLM obtains the storage that it needs for the increased lock entry size from storage for lock table entries or record list entries.

**How to decrease lock entry size:** IRLM does not automatically rebuild if the number of members decreases. To decrease the lock entry size, you must:

1. Quiesce all members of the group, using the following command:
   - `DB1A STOP DB2 MODE (QUIESCE)`

2. If IRLM is not automatically stopped along with DB2, enter the z/OS command:
STOP irlmproc

3. Force all connections to the lock structure to disconnect by issuing the following z/OS command:
   SETXCF FORCE, CONNECTION, STRNAME= strname, CONNAME= ALL

4. Force the deallocation of the lock structure by issuing the following z/OS command:
   SETXCF FORCE, STRUCTURE, STRNAME= strname

5. Change the lock entry size for at least one IRLM. (You should change the value for all of them.)
   If you change the IRLM startup procedure directly, the parameter you change is called MAXUSR5. See the description of MAXUSR5 in the description of the command START irlmproc in DB2 Command Reference. The value of LOCK ENTRY SIZE is translated during the DB2 installation or migration process. The value you enter on the parameter directly is not the same as the value you put in the LOCK ENTRY SIZE parameter of DSNTIPJ.

6. Start the member and IRLM that have the updated value. (You must start the updated member first.)

7. Start all other members.

A group restart occurs when you restart the members. Because you quiesced work before changing the lock entry size, the group restart should be relatively quick. Nonetheless, decreasing the lock entry size is a disruptive procedure. Consider doing this only in situations when the lock entry size is set too high for the number of members in the group, and you cannot alleviate the false contention within the storage boundaries you have.

Reducing the time needed to resolve contentions

When contention exists on a hash class, z/OS uses XCF messages to resolve the conflict. This is how it determines which specific resources are involved in the contention, or if the contention is false. For speedy resolution of contention situations, ensure no queuing of messages exists for XCF message buffers. You can use the XCF Activity Report of RMF to detect this queuing. See z/OS MVS Setting Up a Sysplex for more information about tuning the XCF message buffers.

Avoiding partition locks on all table space partitions

In a partitioned table space, locks are obtained at the partition level. Individual partitions are locked as they are accessed. This locking behavior enables greater concurrency. Partition locks are always acquired, even if the table space was defined in a version of DB2 prior to Version 8 with the LOCKPART NO clause. For table spaces that are defined with LOCKPART NO, DB2 no longer locks the entire table space with one lock when any partition of the table space is accessed.

Each locked partition is a separate parent lock. Therefore, DB2 and IRLM can detect when no inter-DB2 read/write interest exists on that partition and thus do not propagate child L-locks unnecessarily.

See Part 5 (Volume 2) of DB2 Administration Guide for more information about the LOCK TABLE statement.

Restrictions: If any of the following conditions are true, DB2 cannot selectively lock the partitions. It must lock all the partitions:
- The plan is bound with ACQUIRE(ALLOCATE).
- The table space is defined with LOCKSIZE TABLESPACE.
- LOCK TABLE IN EXCLUSIVE MODE is used (without the PART option).
Figure 48 shows partition locks. A partition lock is taken only when the partition is accessed.

![Partitioned Table Space with LOCKPART YES](image)

**Partition locks**

*The duration of a partition lock:* Partition locks follow the same rules as table space locks, and all partitions are held for the same duration. Thus, if one package is using RELEASE(COMMIT) and another is using RELEASE(DEALLOCATE), all partitions use RELEASE(DEALLOCATE). A partition lock can be held past commit point if it uses CURSOR WITH HOLD.

*The mode of a partition lock:* Partition locks have the same possible states as table space locks (IS, IX, S, U, SIX, and X).

*Lock escalation:* Lock escalation occurs when the number of locks per table space exceeds the threshold specified at installation or on the LOCKMAX clause of CREATE or ALTER TABLESPACE. Lock counts are not kept on a partition basis. When the maximum number of locks for a table space is reached, the locks on all partitions are escalated to S or X. Partitions that have not yet been locked are not affected by lock escalation; they remain unlocked. Any partitions already holding S, U, or X locks remain unchanged.

After lock escalation occurs, any unlocked partitions that are subsequently accessed use a gross lock.

*Monitoring selective partition locking:* The performance trace and the DISPLAY DATABASE command give information about selective partition locking.

Use performance trace class 6 (IFCID 0020) to determine whether you have taken advantage of selective partition locking for your partitioned table spaces. If you are using OMEGAMON, you can see this in the locking summary portion of the SQL Activity Report.

**Tuning deadlock and timeout processing**

This section describes how deadlock detection and resource timeouts work in a data sharing environment. This information can help you choose the appropriate times for deadlock detection intervals and for determining a resource timeout value.

This section includes the following information:

- "Global deadlock processing"
- "Global timeout processing" on page 228
- "Recommendations" on page 229

**Global deadlock processing**

In a data sharing environment, deadlocks can occur between transactions on different members. The term *global deadlock* refers to the situation where two or
more members are involved in the deadlock. *Local deadlock* refers to the situation where all of the deadlocked transactions reside on a single member.

**Controlling deadlock detection:** Use the DEADLK parameter in the IRLM startup procedure to control how often IRLM does its deadlock detection processing. Specify the parameter as follows:

\[
\text{DEADLK=}'x,y'\]

- \(x\) The number of seconds between two successive scans for a local deadlock (DEADLOCK TIME value on installation panel DSNTIPJ). The default is 5 seconds. Values can range from 1 to 5 seconds, or 100 to 5000 milliseconds.

- \(y\) The number of local scans that occur before a scan for global deadlock starts (DEADLOCK CYCLE value on install panel DSNTIPJ). IRLM always uses a value of 1.

Global deadlock detection requires the participation of all IRLM members in the data sharing group. Each IRLM member has detailed information about the locks that are held and are being waited for by the transactions on its associated DB2 member. However, to provide global detection and timeout services, each IRLM is informed of all requests that are awaiting globally so that the IRLM can provide information about its own blockers. That IRLM also provides information about its own waiters. The IRLM members use XCF messages to exchange information so that each member has this global information.

**The global deadlock manager:** To coordinate the exchange of information, one IRLM member assumes the role of the global deadlock manager. As IRLM members join or leave the group, the global deadlock manager might change.

**The local deadlock detector:** Each IRLM member in the group must participate in the global deadlock detection process. Each IRLM member (including the one designated as the global deadlock manager) has the role of local deadlock detector.

**Relationship between local and global deadlock detection:** Four XCF messages are required to gather and communicate the latest information from the local deadlock detectors:

1. The local deadlock detector sends its information about lock waiters to the global deadlock manager.

2. The global deadlock manager takes that information from all local deadlock detectors and sends messages to each of the IRLMs in the group. (Because the global deadlock manager is also a local deadlock detector, it receives the same information, although somewhat quicker than the rest of the IRLMs.)

3. Each local deadlock detector checks the global view of resources and determines if it has blockers for other waiters. It passes that information along to the global deadlock manager with its list of waiters.

4. The global deadlock manager, from the information it receives from the local deadlock detectors, determines if a global deadlock or timeout situation exists. If a global deadlock situation exists, DB2 chooses a candidate for the deadlock. The global deadlock manager also determines if any timeout candidate is blocked by an incompatible waiter or holder and, if so, presents that candidate to the owning IRLM, along with any deadlock candidates belonging to that IRLM. When DB2 receives this information, it determines if it should request that IRLM reject any given timeout candidate waiter.

These four messages represent one global detection cycle, which usually takes two to four \(x\)-second intervals to complete (where \(x\) is the number of local cycles).
Figure 49 illustrates an example in which the deadlock time value is set to 5 seconds.

Deadlock time = 5 seconds:

<table>
<thead>
<tr>
<th>Time (seconds)</th>
<th>Local detector sends all candidate waiters</th>
<th>Local detector collects data</th>
<th>Local detector presents victims</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Global deadlock detection cycle

Deadlock detection might be delayed if any of the IRLMs in the group encounter any of the following conditions:
- XCF signalling delays
- IRLM latch contention (can be encountered in systems with extremely high IRLM locking activity)
- A large number of global waiters

Global timeout processing

Just as in a non-data-sharing environment, DB2 calculates the timeout period based on the RESOURCE TIMEOUT and DEADLOCK TIME installation parameter values. DB2 calculates the timeout period as follows:
1. Divide RESOURCE TIMEOUT by DEADLOCK TIME
2. Round to the next largest integer
3. Multiply that integer by DEADLOCK TIME

As described in Part 5 (Volume 2) of DB2 Administration Guide, in non-data-sharing systems, the actual time that a transaction waits on a lock before timing out varies between the timeout period and the timeout period plus one DEADLOCK TIME interval.

For example, if the timeout period for a given transaction is 60 seconds and the DEADLOCK TIME value is 5 seconds, the transaction waits between 60 and 65 seconds before timing out, with the average wait time of 62.5 seconds. This is because timeout is driven by the deadlock detection process, which is activated on a timer interval basis.

Elapsed time until timeout, non-data-sharing: The actual time a process waits until timing out usually falls within the following range:
- MIN LOCAL TIMEOUT = timeout period
- MAX LOCAL TIMEOUT = timeout period + DEADLOCK TIME value
- AVERAGE LOCAL TIMEOUT = timeout period + DEADLOCK TIME value/2

However, the maximum or average values can be larger, depending on the number of waiters in the system or if a heavy IRLM workload exists.

Elapsed time until timeout, data sharing: In a data sharing environment, because the deadlock detection process sends inter-system XCF messages, a given
transaction typically waits somewhat longer before timing out than in a
non-data-sharing environment. How much longer a transaction waits depends on
where in the global deadlock detection cycle that the timeout period actually
expired. However, the length of time a process waits until timing out generally
falls within the following range:

\[
\text{MIN GLOBAL TIMEOUT} = \text{timeout period} + \text{DEADLOCK TIME value} \\
\text{MAX GLOBAL TIMEOUT} = \text{timeout period} + 4 \times \text{DEADLOCK TIME value} \\
\text{AVERAGE GLOBAL TIMEOUT} = \text{timeout period} + 2 \times \text{DEADLOCK TIME value}
\]

Again, the maximum or average values might be larger.

Recommendations
Quick detection of deadlocks and timeouts is necessary in a data sharing
environment to prevent a large number of waiters on each system. A large
numbers of waiters can cause much longer wait times for timeouts and deadlocks.
The following are two recommendations to help prevent a large number of waiters
from developing on each system:

- If your non-data-sharing DB2 subsystem has a problem with deadlocks, consider
  reducing the deadlock time to prevent a long lists of waiters from developing. (If
  you do not have a problem with deadlocks, you most likely will not have to
  change any parameters for data sharing.)
- If you have stringent timeout limits that must be honored by DB2, consider
  decreasing the deadlock time before moving to data sharing, as illustrated in this
  example:

Assume that you have set the timeout period for your non-data-sharing DB2
subsystem to 55 seconds because you want the wait time for timeout to be at or
before 60 seconds. (This assumes that your deadlock time value is five.) In a
data sharing environment, reduce the value of DEADLOCK TIME so that the
timeout period is 40 seconds. This makes it more likely that your actual wait
time for timeouts is at or before 60 seconds.

Monitoring DB2 locking
With data sharing, it is essential to control the volume of global lock requests that
are propagated to the coupling facility and to control the amount of lock
contention, both real and false. You must monitor both the amount and type of
locking that your applications are doing, and you must also make sure that any
locking problems are not caused by data sharing resources, such as an undersized
lock structure, or the overuse of the coupling facility or coupling facility channels
(links).

This section describes the following ways to monitor locking activity and lock
structure activity:

- “Using the command DISPLAY DATABASE” on page 230
- “Using the coupling facility structure activity report of RMF” on page 230
- “Using the DB2 statistics trace” on page 231
- “Using the DB2 accounting trace” on page 232
- “Using the DB2 performance trace” on page 233

The z/OS command D XCF,STRNAME can also be used to monitor lock structure
activity and is described in “Displaying information about specific structures” on
page 152.
Using the command DISPLAY DATABASE

General-use Programming Interface

Use the LOCKS ONLY option on DISPLAY DATABASE to display information about page set, partition, or table locks that are held on resources. The “lock” column of the display describes the type and duration of locks used by corresponding agents.

Figure 50 is an example of output of DISPLAY DATABASE for a table space. The application identified as LSS001 on member DB1A has locked partitions 1 and 2. LSS002 on member DB2A has locked partitions 1 and 3. Partition 4, which has no locks, is not on the display because the ONLY option of DISPLAY DATABASE was used.

Figure 50. Example DISPLAY DATABASE LOCKS for a table space

End of General-use Programming Interface

Using the coupling facility structure activity report of RMF

The Coupling Facility Activity Report of RMF describes activity to all structures in the coupling facility for a given time period. Figure 51 shows a partial report, giving information about:

- A: Total number of lock-related requests.
- B: Number of requests that were deferred because of contention.
- C: The number of deferred requests that were caused by false contention.

Figure 51. Partial RMF Coupling Facility Activity report for lock structure

Determining the contention percentages: Use the following calculations to determine the percentage of global contention and false contention:
• Total contention is the number of deferred requests (B) divided by the total number of requests (A), multiplied by 100. So, for this example:

\[
\frac{621}{162000} \times 100 = 0.387\%
\]

This indicates that the global contention rate is approximately 0.39% (a good figure).

• False contention is the number of false contentions (C) divided by the total number of requests (A) multiplied by 100. For this example:

\[
\frac{212}{162000} \times 100 = 0.13\%
\]

Thus, the rate of false contention is 0.13% (a very good figure).

**Using the DB2 statistics trace**

The DB2 statistics trace provides counters that track the amount of global locking activity and contention that each member in the data sharing group is encountering. This trace runs with low overhead. Keep the DB2 statistics trace turned on to allow continuous monitoring of each subsystem.

See Figure 52 to view the type of information provided by a statistics trace.

<table>
<thead>
<tr>
<th>DATA SHARING LOCKING</th>
<th>QUANTITY /SECOND</th>
<th>THREAD /SECOND</th>
<th>COMMIT /SECOND</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBAL CONTENTION RATE (%)</td>
<td>3.24 A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L-LOCKS XES RATE (%)</td>
<td>46.95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOCK REQUESTS (P-LOCKS)</td>
<td>147.8K 246.42</td>
<td>N/C 1.61</td>
<td></td>
</tr>
<tr>
<td>UNLOCK REQUESTS (P-LOCKS)</td>
<td>147.8K 246.38</td>
<td>N/C 1.61</td>
<td></td>
</tr>
<tr>
<td>CHANGE REQUESTS (P-LOCKS)</td>
<td>0.00 0.00</td>
<td>N/C 0.00</td>
<td></td>
</tr>
<tr>
<td>SYNCH.XES - LOCK REQUESTS</td>
<td>779.1K 1298.60</td>
<td>N/C 8.47</td>
<td></td>
</tr>
<tr>
<td>SYNCH.XES - CHANGE REQUESTS</td>
<td>253.8K 423.05</td>
<td>N/C 2.76</td>
<td></td>
</tr>
<tr>
<td>SYNCH.XES - UNLOCK REQUESTS</td>
<td>922.9K 1538.25</td>
<td>N/C 10.04</td>
<td></td>
</tr>
<tr>
<td>ASYNCH.XES - RESOURCES</td>
<td>81690.00 136.16</td>
<td>N/C 0.89</td>
<td></td>
</tr>
<tr>
<td>SUSPENDED - IRLM GLOBAL CONT</td>
<td>1871.00 3.12</td>
<td>N/C 0.02</td>
<td></td>
</tr>
<tr>
<td>SUSPENDED - XES GLOBAL CONT.</td>
<td>59745.00 99.58</td>
<td>N/C 0.65</td>
<td></td>
</tr>
<tr>
<td>SUSPENDED - FALSE CONTENTION</td>
<td>3964.00 6.61</td>
<td>N/C 0.04</td>
<td></td>
</tr>
<tr>
<td>INCOMPATIBLE RETAINED LOCK</td>
<td>0.00 0.00</td>
<td>N/C 0.00</td>
<td></td>
</tr>
<tr>
<td>NOTIFY MESSAGES SENT</td>
<td>1646.00 2.74</td>
<td>N/C 0.02</td>
<td></td>
</tr>
<tr>
<td>NOTIFY MESSAGES RECEIVED</td>
<td>2359.00 3.93</td>
<td>N/C 0.03</td>
<td></td>
</tr>
<tr>
<td>P-LOCK/NFY EX.ENGINES UNAVAIL</td>
<td>500.00 N/A</td>
<td>N/A N/A</td>
<td></td>
</tr>
<tr>
<td>P-LCK/NFY EX.ENGINES UNAVAIL</td>
<td>0.00 0.00</td>
<td>N/C 0.00</td>
<td></td>
</tr>
<tr>
<td>PSET/PART P-LCK NEGOTIATION</td>
<td>0.00 0.00</td>
<td>N/C 0.00</td>
<td></td>
</tr>
<tr>
<td>PAGE P-Lock NEGOTIATION</td>
<td>1137.00 1.90</td>
<td>N/C 0.01</td>
<td></td>
</tr>
<tr>
<td>OTHER P-Lock NEGOTIATION</td>
<td>0.00 0.00</td>
<td>N/C 0.00</td>
<td></td>
</tr>
<tr>
<td>P-Lock CHANGE DURING NEG.</td>
<td>1137.00 1.90</td>
<td>N/C 0.01</td>
<td></td>
</tr>
</tbody>
</table>

Figure 52. Data sharing locking block of OMEGAMON statistics trace

Table 52 contains an explanation of the fields that are shown in Figure 52.

**Table 52. Explanation of the fields in a OMEGAMON statistics trace**

<table>
<thead>
<tr>
<th>Field</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>The global contention rate.</td>
</tr>
<tr>
<td>B</td>
<td>These counters indicate the total number of lock, change, and unlock requests (including L-locks and P-locks) that were propagated to XES synchronously.</td>
</tr>
</tbody>
</table>
### Table 52. Explanation of the fields in a OMEGAMON statistics trace (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C</strong></td>
<td>The number of resources (including L-locks and P-locks) that were propagated to XES asynchronously. DB2 uses the term <em>asynchronous</em> to mean that the request was done under a system execution unit, asynchronous to the allied work unit. This particular counter can be incremented when, for example, one member has an IS lock on a particular table space and another member requests an IX lock. The S child locks held by the first member must be propagated under a system execution unit to XES and the coupling facility. See <a href="#">Figure 46 on page 221</a> for an example of this. It is possible for these asynchronous child lock propagations to encounter false contention. If so, the false contention is counted in RMF statistics, but not in DB2.</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>The number of real contentions, as detected by IRLM.</td>
</tr>
<tr>
<td><strong>E</strong></td>
<td>The number of real contentions, as detected by XES, that were not IRLM-level contentions. IRLM has knowledge of more lock types than XES. Thus, IRLM often resolves contention that XES cannot. The most common example of XES-level contention is usually the intent locks (IS and IX) on the parent L-locks. IS and IX are compatible to IRLM but not to XES. Another common example is the U and S page L-locks; U and S are compatible to IRLM, but not to XES.</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td>The number of false contentions.</td>
</tr>
</tbody>
</table>

### Calculating global contention percentages:

To calculate the global contention percentages, use the statistics trace shown in [Figure 52 on page 231](#). These calculations only account for synchronous lock requests.

- Total contention is the total number of suspends because of contention (D + E + F) divided by the total number of requests that went to XES (excluding asynchronous requests) (three fields under B) + (D + E + F) multiplied by 100. For this example:
  
  \[
  \frac{65580}{2021380} \times 100 = 3.24\% 
  \]
  
  This indicates that the contention rate is approximately 3.24 percent (A). Because this is such a low rate of contention, it is probably not necessary to determine the amount of false contention. However, knowing how to calculate false contention can be useful.

- False contention is the number of false contentions (F) divided by the total number of requests that went to XES (excluding asynchronous requests) (three fields under B) + (D + E + F) multiplied by 100. For this example:

  \[
  \frac{3964}{2021380} \times 100 = .20\% 
  \]

  Thus, the approximate rate of false contention is .2 percent, a very low figure.

### Using the DB2 accounting trace

Use the DB2 accounting trace to determine which users or plans are experiencing global lock contention. The accounting trace provides a summary of thread resource usage within DB2. DB2 threads experiencing global lock contention are shown in accounting trace class 1, as shown in [Figure 53 on page 233](#). The accumulated elapsed time of the suspensions are shown in accounting trace class 3.
Calculating false contention percentages: To calculate the false contention percentages, use the accounting trace shown in Figure 53. False contention is the total number of suspends due to false contention (\( C + D + E \)) divided by the total number of requests that went to XES (\( A \)), multiplied by 100.

So, for this example:

\[(3 / 457) \times 100 = .66\%\]

False contention is approximately 0.66% of the total number of requests that went to XES.

Measuring transaction locking optimizations: To see how well global transaction locking optimizations are working in your application, you must:

1. Determine the total number of L-lock LOCK requests propagated to XES (\( X \) in the following example) by subtracting P-lock LOCK requests from the total number of LOCK requests propagated to XES:

\[ C - B = X \]

In this example:

\[ 263 - 9 = 254 \]

2. Divide the total number of L-lock LOCK requests that were propagated to XES (\( X \)) by the total number of LOCK requests (\( A \)).

In this example:

\[ 254 / 327 = .77 \]

3. Multiply by 100.

In this example:

\[ .77 \times 100 = 77\% \]

For this particular work load, with 100% of the data of inter-DB2 read/write interest, 23% of LOCK requests were not propagated to XES and the coupling facility because of locking optimizations.

Using the DB2 performance trace

The DB2 performance trace gives more detailed information about which shared resources are experiencing contention. Performance traces are generally activated on an as needed basis because of their added overhead. Performance trace class 6 (specifically, IFCID 0045) indicates whether the suspension is because of contention.
This trace causes DB2 to write a trace record every time a lock is suspended and every time it is resumed. Part of the data that is recorded is the resource name that is experiencing the contention. By determining which shared resources are experiencing the lock contention, you might be able to make some design changes or other tuning actions to increase concurrency. Check for contention within IRLM, because IRLM indicates true contention over resources.

**Changing the size of the lock structure**

This section describes two possible ways to change the size of the lock structure. You can change the size dynamically or by rebuilding the structure after making a CFRM policy change. When choosing which way to change the size, consider the level of the coupling facility and when you want the storage of the lock table portion of the lock structure changed.

When you change the size of the lock structure dynamically, only the modify lock list portion of the structure is changed immediately. The size of the lock table portion remains unchanged until the structure is rebuilt. When the structure is rebuilt, the structure is reallocated at the changed size with the storage divided between the lock table and the record table based on your IRLMPROC LTE value or the value that is set using the MODIFY irlmproc,SET,LTE= command, if either of these values is nonzero. (As described in "Coupling facility structure size allocation" on page 47 if the structure was forced with the SETXCF FORCE command, it is reallocated at the INITSIZE that is specified in the CFRM policy and not the changed size.)

**Changing the lock structure size dynamically**

If all of the following conditions are true, you can use the SETXCF START,ALTER command to change the lock structure size:

- The lock structure is allocated in a coupling facility with CFLEVEL greater than zero.
- The currently allocated size of the structure is less than the maximum size that is defined in the SIZE parameter of the CFRM policy.

Enter the following command to change the lock structure size:

```
SETXCF START,ALTER,STRNAME=DSNDB0A_LOCK1,SIZE=newsize
```

This example assumes that newsize is less than or equal to the maximum size defined for the lock structure and that the group name is DSNDB0A. If the maximum size (SIZE in the CFRM policy) is still not big enough, you must increase the lock storage in the CFRM policy and rebuild the lock structure.

**Important:** For a duplexed lock structure, you are changing the size of both the primary and secondary structure with a single command.

Because the dynamic method affects only the record table entries portion of the lock structure, the impact of changing the lock size can have a disproportionately large effect on the record table list portion of the structure. For example, if you halve the size of the lock structure, it can result in all of the available record table entries being taken away—probably not the result you want. For the same reason, if you double the lock structure size, the increased storage space is used entirely by the record table unless a rebuild occurs or the group is shut down, the structure is forced, and the group is restarted. If either of these are done after the size is changed, the split of the new structure is determined by the number of lock table entries requested by the first IRLM to connect to the group.
Changing the size of the lock structure by rebuilding

If any of the following conditions are true, you must rebuild the lock structure in order to change its size:

- The lock structure is allocated in a coupling facility at CFLEVEL=0.
- The allocated size of the structure is already at the maximum size defined by the SIZE parameter of the CFRM policy, and you need to increase the maximum limit.
- You want to change the size of the lock table portion of the lock structure.

You must do at least one of the following procedures to rebuild the lock structure with a new size:

1. If you are at maximum size or want to increase your maximum size available, change the CFRM POLICY SIZE to the desired size. If you are satisfied with the number of lock table entries that you have been getting, leave the INITSIZE the same. If you want more lock table entries to decrease your contention rate, change the INITSIZE to accommodate the added number of entries, by doing either of the following:
   a. If you are allowing IRLM to determine how many LTE entries to request, ensure that when you change INITSIZE, it remains an even power of 2 so that there is a 1:1 split between the lock table storage and the record table storage. For example, if your current INITSIZE is 16 MB, increase it to 32 MB.
   b. If you are controlling the number of lock table entries by specifying LTE= in the IRLMPROC or by issuing the MODIFY irlmproc,SET,LTE= command, your INITSIZE does not need to be a power of 2, but it is still recommended. You do need to ensure that it is large enough to accommodate the storage required for your LTE= value and still large enough to create sufficient record table entries to handle your update volume.

2. If you want to change the size of the lock table:
   a. If your contention rates are low and you want fewer lock table entries, do the following procedure:
      1) If you are letting IRLM control the number of lock entries to request, you must decrease the INITSIZE by a power of 2.
         Important: This also decreases the record table size by half.
      2) If you want to control the number of lock table entries to request, issue the MODIFY irlmproc,SET,LTE= command, where the LTE= value reduces the current number of lock entries by half. For example, assume that you currently have 16 MB lock entries, specify:
         MODIFY irlmproc,SET,LTE=8
         This method increases the number of record table entries, because the INITSIZE is not altered.
   b. If your contention rates are high and you want more lock table entries, do the following procedure:
      1) If you are letting IRLM control the number of lock entries to request, you should increase the INITSIZE by a power of 2. This also doubles the size of the record table.
      2) If you want to control the number of lock table entries to request, issue the MODIFY irlmproc,SET,LTE= command, where the LTE= value increases the current number of lock entries by powers of 2. For example, assume that you currently have 8 MB of lock entries, specify:
Important: This method decreases the number of record table entries if the INITSIZE is not altered.

If the INITSIZE is not large enough to provide the lock table storage and adequate record table storage, you will experience failures when trying to write RLE to the coupling facility. If you do not want the record table size to change, you must increase the INITSIZE by the same amount of storage that will be used by the additional lock table entries. For example, assume that you currently have 8 MB of lock table entries and you issue the following command:

```
MODIFY irlmproc,SET,LTE=16
```

With an INITSIZE of 32 MB, you will not have any storage left for record table entries. Determine the additional storage needed by using the following formula:

\[
\text{lock table entries (new)} - \text{lock table entries (old)} \times 2\text{byte} = \text{additional storage for lock entries}
\]

In this example:

\[
16\text{MB} - 8\text{MB} \times 2\text{byte} = 16\text{MB}
\]

So adding 16 MB to the original INITSIZE of 32 MB will result in the new INITSIZE of 48 MB.

Important: For a duplexed lock structure, you are changing the size of both the primary and secondary structure with a single command.

---

### Tuning group buffer pools

This section describes information about how DB2:

- Ensures that DB2 does not read down-level data that is cached in its member buffer pools (cache coherency).
- Enables, as much as possible, a quick refresh of a down-level page without having to go to disk.

With DB2 data sharing, group buffer pools are a key component of cache coherency as are the subsystem locking mechanisms, the P-locks, used in that process. Your understanding of these processes is helpful when tuning the data sharing group for best performance.

With DB2 data sharing, a database page can reside:

- In a local buffer pool
- In a group buffer pool
- On disk

Database pages continue to be cached in each member's buffer pools before they can be referenced or updated. Each sharing member can control its own buffer pool configurations (the size and number of buffer pools). However, if inter-DB2 read/write interest exists in the data, the group buffer pool is also used for caching data (unless the group buffer pool is defined as GBPCACHE (NO) or the page set is defined with GBPCACHE NONE).

The group buffer pool contains information necessary for maintaining cache coherency. Pages of GBP-dependent page sets are registered in the group buffer pool. When a changed page is written to the group buffer pool, all members that
have this page cached in their buffer pools are notified that the page has been invalidated (this notification does not cause a processing interruption on those systems). This is called cross-invalidation. When a member needs a page of data and finds it in its buffer pool, it tests to see if the buffer contents are still valid. If not, then the page must be refreshed, either from the group buffer pool or disk.

This section describes the following topics:
- “Assigning page sets to group buffer pools” on page 238
- “Inter-DB2 interest and GBP-dependency” on page 238
- “Using P-locks” on page 243
- “Read operations” on page 246
- “Write operations” on page 248
- “Group buffer pool thresholds” on page 255
- “Monitoring group buffer pools” on page 258
- “Determining the correct size and ratio” on page 263
- “Changing group buffer pools” on page 269

Assigning page sets to group buffer pools

Any data sharing group can have up to 50, 4-KB page size group buffer pools, and up to ten each of 8-KB, 16-KB, and 32-KB page size group buffer pools. Different group buffer pools can reside in different coupling facilities. The strict naming convention you must use ensures that DB2 can map the group buffer pools to the individual member buffer pools. For example, buffer pool BP0 maps to group buffer pool GBP0. Thus, your choice of buffer pool determines which group buffer pool is used. GBP0 is the default unless you have specified a different default for user data and indexes on installation panel DSNTP1.

For example, to assign table space DSN8S81D to GBP2, you must take the following actions:
1. Stop all access to the table space by issuing the following command:
   -DB1A STOP DATABASE(DSN8S81A) SPACENAM(DSN8S81D)
2. Change the buffer pool assignment by running the following SQL statement:
   ALTER TABLESPACE DSN8S81A.DSN8S81D
   BUFFERPOOL BP2;
3. Allow access to the table space by issuing the following command:
   -DB1A START DATABASE(DSN8S81A) SPACENAM(DSN8S81D)

The above procedure works only when you are altering a table space to a buffer pool with the same page size. For information about changing an existing page size, see Part 2 (Volume 1) of DB2 Administration Guide

Recommendations for performance

- For best performance, keep GBP-dependent page sets in separate buffer pools from non-GBP-dependent page sets. For example, keep work file table spaces, which are always non-GBP-dependent, in different buffer pools than those used by GBP-dependent page sets. Assign work file table spaces to a buffer pool other than BP0.

This separation helps DB2 more efficiently handle registering pages to, and unregistering pages from, the group buffer pool.
Data rows that are too large for a single 4-KB page can use an 8-KB, 16-KB, or 32-KB page to improve the balance for different processing requirements. This allows you to store larger rows more efficiently and improve performance.

### How to keep data from being shared

It is possible, although not necessarily recommended, to restrict access to data to a single member. If you choose to do this, consider the following operational issues:

- You cannot do workload balancing for that data, because the other members of the group are not aware of that data. Thus, the member that has access to the data can become overloaded if access to that data increases over time.
- Availability is compromised, because if the member that owns the data goes down, no other member can access that data.
- You might have to set up special affinities to allow the application access to that data. Work cannot be automatically routed around the group to find the data.

### Defining private data:

If you want access to a table space named NOSHARE limited only to DB2C, you could assign NOSHARE to a previously unused buffer pool, such as BP25, using the ALTER TABLESPACE statement. Do not define a group buffer pool that corresponds to BP25, and assign BP25 a size of zero on any other member of the group. This prevents the other members of the group from attempting to use this buffer pool and therefore prevents the other members from accessing table space NOSHARE.

### Inter-DB2 interest and GBP-dependency

The concepts of inter-DB2 read/write interest and group buffer pool dependency (GBP-dependency) are closely related. Whenever inter-DB2 read/write interest exists on a page set or partition, that object is GBP-dependent. Conversely, if no inter-DB2 read/write interest exists on a page set or partition, the object is usually not GBP-dependent. Sometimes an object still has pages cached in the group buffer pool, and it can remain GBP-dependent even after the inter-DB2 read/write interest has gone away. Table 53 shows how to determine if a page set is GBP-dependent based on the inter-DB2 interest of one member and all other members.

**Table 53. Determining group buffer pool dependency**

<table>
<thead>
<tr>
<th>One member's interest</th>
<th>Other members' interest</th>
<th>Is page set GBP-dependent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read-only</td>
<td>None, Read-only</td>
<td>No</td>
</tr>
<tr>
<td>Read-only</td>
<td>Read/Write</td>
<td>Yes</td>
</tr>
<tr>
<td>Read/Write</td>
<td>None</td>
<td>No ¹</td>
</tr>
<tr>
<td>Read/Write</td>
<td>Read-only</td>
<td>Yes</td>
</tr>
<tr>
<td>Read/Write</td>
<td>Read/Write</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Exception:**

1. The page set remains GBP-dependent for some time before DB2 removes the dependency. DB2 might not be able to remove the GBP-dependency if applications update the page set without issuing periodic commits.

### How DB2 tracks interest

The mechanism that DB2 uses to express interest in an object is a global lock called a *physical lock* (P-lock). These locks are "physical" in contrast to transaction locks, sometimes called "logical locks" (L-locks). The level of P-lock that tracks DB2 read/write interest is a *page set P-lock*. There are also page P-locks, which serve another role.
Although you do not have as much control over physical locks as over transaction locks, P-locks play an important part in how DB2 tracks inter-DB2 interest. Page set and page P-locks are described in more detail in “Using P-locks” on page 243.

Page set P-lock operations occur on each member, and reflect that member’s level of interest in a page set. Even if only one data sharing member is active, page set P-lock operations still occur. Table 54 shows when those operations occur on each member.

<table>
<thead>
<tr>
<th>Event</th>
<th>Page set P-lock operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page set or partition data sets are physically opened.</td>
<td>Page set P-lock is obtained in a read-only state.</td>
</tr>
<tr>
<td>Page set or partition is first updated.</td>
<td>Page set P-lock is changed to a read-write state.</td>
</tr>
<tr>
<td>No update was done within an installation-specified time period or number of checkpoints (read-only switching).</td>
<td>Page set P-lock is changed to a read-only state.</td>
</tr>
<tr>
<td>Page set or partition data sets are closed.</td>
<td>Page set P-lock is released.</td>
</tr>
</tbody>
</table>

In addition to the events mentioned in Table 54, a special case can occur under the following conditions:

- A single member with read/write interest and any number of members with read-only interest exist.
- All members have read-only interest and the page set or partition has been GBP-dependent since the time it was physically opened.

In those conditions, if the read-only members do not reference the page set again in a certain amount of time, DB2 physically closes the page set for the read-only members to remove the inter-DB2 read/write interest. See Figure 55 on page 240 for an example of this.

**Scenarios of P-Lock operations**

Figure 54 on page 240 shows a typical sequence of events for P-locking and P-lock negotiations between two members of a data sharing group.
Figure 54. **P-lock operations between two members.** The arrows indicate that the members are negotiating P-lock modes.

**Figure 55** shows what happens when a single updater remains.

<table>
<thead>
<tr>
<th>DB1G</th>
<th>GBP-dependent?</th>
<th>DB2G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open page set for read</td>
<td>Yes</td>
<td>Update page set</td>
</tr>
<tr>
<td>Take P-lock in S mode</td>
<td>Change P-lock to SIX</td>
<td></td>
</tr>
<tr>
<td>Change P-lock to IS</td>
<td>Read-only switch</td>
<td>Change P-lock to S</td>
</tr>
<tr>
<td>Change P-lock to S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical close</td>
<td>No</td>
<td>Physically close page set</td>
</tr>
<tr>
<td>Release P-lock</td>
<td></td>
<td>Release P-lock.</td>
</tr>
</tbody>
</table>

**Tuning recommendation**

To avoid having DB2 go in and out of GBP-dependency too often, tune the subsystem parameters that affect when data sets are switched to a different state. These parameters are found on installation panel DSNTIPN:

- CHECKPOINT FREQ
- RO SWITCH CHKPTS
- RO SWITCH TIME

See Part 5 (Volume 2) of *DB2 Administration Guide* for more information about these parameters.

**Determining the amount of inter-system sharing**

To determine the amount of inter-system sharing, you can use the DISPLAY BUFFERPOOL command with LIST option to get a snapshot of your objects and how they are being shared across members. Using this command, you will see, by partition, which members have interest in the object, which level of interest they
have (the P-lock state), and which member is the castout owner. You can also use
the DBNAME, SPACENAME, GBPDEP, and CASTOWNR options to limit the
scope of the report.

Issue the command:

-DISPLAY BUFFERPOOL(BP0) LIST(*)

Your output will be similar to the output that is shown in Figure 56.

```
DSNB401I: BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 21
DSNB402I: VIRTUAL BUFFERPOOL SIZE = 500 BUFFERS
          ALLOCATED = 500 TO BE DELETED = 0
          IN-USE/UPDATED = 0
DSNB406I: VIRTUAL BUFFERPOOL TYPE -
          CURRENT = PRIMARY
          PENDING = PRIMARY
          PAGE STEALING METHOD = LRU
DSNB404I: THRESHOLDS -
          VP SEQUENTIAL = 80
          DEFERRED WRITE = 85
          VERTICAL DEFERRED WRT = 80, 0
          PARALLEL SEQUENTIAL = 50
          ASSISTING PARALLEL SEQT= 0
DSNB460I: PAGE SET/PARTITION LIST INFORMATION--------------
          DATA SHARING INFO------
          TS GBP MEMBER CASTOUT USE P-LOCK STATE
          DATABASE SPACE NAME PART IX DEP NAME OWNER COUNT
          ========= =========== = == = = = == ======== = = = = = =
          DSNDB01 DSNLLX02 IX Y DB1AA 0 IS
          DSB1AB Y 0 SIX
          DSNB01 DSN8S61E 001 TS Y DB1AA 0 IS
          DSB1AB Y 0 SIX
          002 TS N DB1AA 0 S
          DSB1AB 0 S
          DSNB01 DSN8S61A DSN8S61E 001 TS Y DB1AA 0 IS
          DSB1AB Y 0 SIX
          002 TS N DB1AA 0 S
          DSB1AB 0 S
          SPT01 TS N DB1AA 0 S
          DSB1AB 0 S
          DSNB07 DSN4K01 TS N DB1AA 0 S
          ...
DSNB460I: DSNB1CMD '-DIS BPOOL' NORMAL COMPLETION
```

Figure 56. Sample DISPLAY BUFFERPOOL output showing the amount of inter-system
sharing.

Displaying GBP-dependent page sets

--- General-use Programming Interface ---

To determine if a particular page set is GBP-dependent, use the DISPLAY
DATABASE command with the LOCKS option:

-DB1A DISPLAY DATABASE(DSN8B1A) SPACE(DSN8B1D) LOCKS

Your output will be similar to the output that is shown in Figure 57 on page 242.
Page set P-locks are identified by a member name rather than a correlation ID. They are also identified by 'PP' as the lock unit. If any of the P-locks shown in the output have a lock state of NSU, SIX, or IX, the identified page set is GBP-dependent. Thus, the output in Figure 57 shows that DSN8S81D is GBP-dependent.

**Figure 57. Sample DISPLAY DATABASE output showing GBP-dependent table spaces**

---

**Determining GBP-dependency for a particular member:** At times you might need to know what the impact is to bring down a particular member or to disconnect a particular member from the group buffer pool. You can use the DISPLAY BUFFERPOOL command with the GBPDEP(Y) option to discover whether a particular member has any page sets opened that are GBP-dependent:

```
-DB1A DISPLAY BUFFERPOOL(BP0) GBPDEP(Y)
```

Your output will be similar to the output that is shown in Figure 58.

0DIS BPOOL(BP0) GBPDEP(Y)
DSNB400I @ BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 21
DSNB402I @ VIRTUAL BUFFERPOOL SIZE = 500 BUFFERS
  Allocated = 500 TO BE DELETED = 0
  IN-USE/UPDATED = 0
DSNB406I @ VIRTUAL BUFFERPOOL TYPE -
  CURRENT = PRIMARY
  PENDING = PRIMARY
  PAGE STEALING METHOD = LRU
DSNB404I @ THRESHOLDS -
  VP SEQUENTIAL = 80
  DEFERRED WRITE = 85
  VERTICAL DEFERRED WRT = 80,0
  PARALLEL SEQUENTIAL = 90
  ASSISTING PARALLEL SEQT= 0

**Figure 58. Sample DISPLAY BUFFERPOOL output indicating page sets and partitions are group-buffer-pool-dependent**
Using P-locks

This section describes a type of lock specific to data sharing called a physical lock (P-lock). In particular, it describes P-locks on page sets and on pages, and it discusses how to monitor and tune those locks.

Page set P-locks

P-locks are used on physical objects stored in buffers (table spaces, index spaces, and partitions). P-locks have complete partition independence: it is possible to have a P-lock on one partition of a page set and not on another. A P-lock on a page set does not necessarily mean that a P-lock exists on the corresponding index.

P-locks do not control concurrency but they do help DB2 track the level of interest in a particular page set or partition and determine the need for cache coherency controls.

P-locks differ from L-locks (transaction locks) in the following ways:

- P-locks are owned by a member. After a P-lock is obtained for the subsystem, later transactions accessing the locked data do not have to incur the expense of physical locking.
- The mode of a P-lock can be negotiated. If one member requests a P-lock that another member holds in an incompatible mode, the existing P-lock can be made less restrictive. The negotiation process usually involves registering pages or writing pages to the group buffer pool, and then downgrading the P-lock to a mode that is compatible with the new request.

Displaying retained P-locks

Just as with transaction locks, certain P-locks can be retained because of a system failure. A retained P-lock means that other members cannot access the data that the P-lock is protecting if the accessing member requests a P-lock in an incompatible state. Thus, if a member fails holding an IX page set P-lock, it is still possible for another member to obtain an IX page set P-lock on the data. See "Active and retained locks" on page 183 for more information about retained locks and when they are released.

Use the DISPLAY DATABASE command with the LOCKS option to determine if retained locks exist on a table space, index, or partition. An “R” in the LOCKINFO column indicates that a lock is retained.

Table 55 shows the possible modes of access for a page set and the P-lock state that is retained if the member that is represented in the first column fails.

<table>
<thead>
<tr>
<th>One member's interest</th>
<th>Other members' interest</th>
<th>Retained P-lock states of single member</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read-only</td>
<td>None, Read-only</td>
<td>None</td>
</tr>
<tr>
<td>Read-only</td>
<td>Read/Write</td>
<td>None</td>
</tr>
<tr>
<td>Read/Write</td>
<td>None</td>
<td>X or NSU ¹</td>
</tr>
<tr>
<td>Read/Write</td>
<td>Read-only</td>
<td>IX ²</td>
</tr>
<tr>
<td>Read/Write</td>
<td>Read/Write</td>
<td>IX</td>
</tr>
</tbody>
</table>
Table 55. Determining retained P-lock state (continued)

<table>
<thead>
<tr>
<th>One member's interest</th>
<th>Other members' interest</th>
<th>Retained P-lock states of single member</th>
</tr>
</thead>
</table>

Notes:
1. NSU stands for “non-shared update”. It acts like an X lock, but is only used during P-lock negotiation from an X to an SIX.
2. The P-lock is retained in SIX mode if the page set or partition is an index that is not on a DB2 catalog or directory table space.

Page P-locks
At times, a P-lock must be obtained on a page to preserve physical consistency of the data between members. These locks are known as page P-locks. Page P-locks, are used, for example, when two subsystems attempt to update the same page of data and row locking is in effect. These locks are also used for GBP-dependent space map pages and GBP-dependent leaf pages for indexes, regardless of locking level. Page P-locks can also be retained if a member fails.

If an index page set or partition is GBP-dependent, DB2 does not use page P-locks for that index page set or partition if all of the following are true:
• Only one member is updating the index page set or partition
• No repeatable read claimers exist on the read-only members for the index page set or partition
• The index is not on a DB2 catalog or directory table space

Because of the possible increase in P-lock activity with row locking, evaluate row locking carefully before using it in a data sharing environment. If you have an update-intensive application process, the amount of page P-lock activity might increase the overhead of data sharing.

To decrease the possible contention on those page P-locks, consider using page locking and a MAXROWS value of one on the table space to simulate row locking. You can get the benefits of row locking without the data page P-lock contention that comes with it. A new MAXROWS value does not take effect until you run REORG on the table space.

Monitoring P-locks
More overhead exists when inter-DB2 read/write interest exists in a page set. Although DB2 does dynamically track inter-DB2 read/write interest, which helps avoid data sharing overhead when it is not needed, you will pay some cost for the advantages of data sharing.

Monitoring P-lock activity, especially page P-locks, can help you determine if you need to take steps to control inter-DB2 read/write interest. If excessive global contention exists that cannot be resolved by any tuning measures, you might need to reduce the locking overhead by keeping some transactions and data together on a single system.

How to find information about page set P-locks: You can use the DISPLAY DATABASE command with the LOCKS option to find information about page set P-locks, including which member is holding or waiting for P-locks, and whether P-locks are being held because of a DB2 failure. Figure 57 on page 242 has sample output obtained from the command. A “PP” in the LOCKINFO field of the output indicates that a particular lock is a page set or partition P-lock.
You can also obtain information about P-locks, along with information about transaction locking, from the statistics and accounting traces. Performance class 20 (IFCID 0251) also contains information about P-lock negotiation requests. IFCID 0251 is mapped by DSNDQW04.

**How to find information about page P-locks:** Page P-locking activity is recorded, along with the rest of the data sharing locking information, in the statistics and accounting trace classes. You can find more detail about those page P-locks in performance trace class 21 (IFCID 0259). IFCID 0259 allows you to monitor page P-locking without having to use a full DB2 lock trace. IFCID 0259 is mapped by DSNDQW04.

**Reducing space map page contention**

This section describes a couple of options you can use when defining table spaces to help reduce the space map hot spots that can occur when a lot of update, insert, or delete activity occurs on a page set from multiple members of a group. The MEMBER CLUSTER option can reduce contention when doing heavy sequential inserts, and the TRACKMOD NO option can reduce contention when any type of insert, update, or delete activity occurs.

Understand the implications of each option before choosing either one, as both options have drawbacks.

**Member affinity clustering:** For applications that do heavy sequential insert processing from multiple members, the contention on the space map or for the data pages at the end of the table can be considerable. The MEMBER CLUSTER option of CREATE TABLESPACE causes DB2 to manage space for inserts on a member-by-member basis instead of by using one centralized space map. Table spaces defined with MEMBER CLUSTER have the following characteristics:

- Data that is inserted by the SQL INSERT statement is not clustered by the implicit clustering index (the first index) or the explicit clustering index.
- DB2 chooses where to locate the data in such a way that avoids lock and latch contention. In general, it tries to insert data in a place that is covered by the locally cached space map page. If it cannot find space there, it continues to search through space map pages until it can find a place for which the space map page is available. As a result, space in a data set might not be fully used. But when the data set reaches the maximum number of extents, lock contention can increase and DB2 does use the entire space.
- Each space map covers 199 data pages. Because there are more space map pages and some might be partially used, table spaces that are defined with MEMBER CLUSTER can use more disk.
- To reduce the overhead of reacquiring page P-locks, a page P-lock is held longer for MEMBER CLUSTER table spaces.

The downside to using MEMBER CLUSTER is that data is not inserted in clustering order. If you have a query application that performs best when data is in clustering order, you should run REORG on the table space before starting the query application.

**Avoid tracking updates:** The TRACKMOD NO option of CREATE or ALTER TABLESPACE can reduce coupling facility overhead caused by constant updating of the space map pages of the page sets. With TRACKMOD NO, DB2 does not keep track of changed pages in the space map page of the page set. By choosing TRACKMOD NO and not tracking updates, less coupling facility overhead exists,
but the cost of incremental image copies is much higher because DB2 must use a
table space scan to read all pages to determine if the page has been changed and
thus needs to be copied.

If longer copy times means you must take fewer incremental copies, monitor your
active log data sets to ensure that you do not have to go to a tape archive data set
to do recovery. You might need to make the active log data sets larger, specify
more active log data sets, or archive to disk to avoid this possibility.

**Recommendation:** If you rarely or never use incremental image copies, or if you
always use DFSMS concurrent copy with DB2 LOGONLY recovery, use
TRACKMOD NO.

**Read operations**

This section describes how the process of reading data is changed for data sharing.

**Where DB2 looks for a page**

DB2 searches for pages in this order:

1. In the local buffer pool. If the page is invalid, DB2 refreshes the page from the
group buffer pool (or disk).
2. In the group buffer pool. DB2 checks the group buffer pool for a page if the
page set is defined as GBPCACHE ALL or if the page set or the partition is
GBP-dependent, unless one of the following conditions is true:
   - Group buffer pool is defined as GBPCACHE(NO).
   - Page set is defined as GBPCACHE NONE.
   - Page set is defined as GBPCACHE SYSTEM and the page being read is not a
     space map page.
3. On disk. If the page is not in the group buffer pool, DB2 refreshes the page in
   the buffer pool from disk.

For duplexed group buffer pools, read activity occurs only against the primary
structure.

**Testing the page validity**

Part of the process of controlling cache coherency is testing to see if a page that is
referenced in the buffer pool must be refreshed from the group buffer pool or disk
because it might no longer be the most current version of the data. This is known
as testing the page *validity*. Because DB2 tracks the level of interest in a page set
across the group, it is not always necessary to make this test. Table 56 indicates
when this test is performed.

For duplexed group buffer pools, only the primary structure is used for
cross-invalidations.

*Table 56. Determining when page validity must be tested*

<table>
<thead>
<tr>
<th>One member's interest</th>
<th>Other members' interest</th>
<th>Test page validity?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read-only</td>
<td>None, Read-only</td>
<td>No</td>
</tr>
<tr>
<td>Read-only</td>
<td>Read/Write</td>
<td>Yes</td>
</tr>
<tr>
<td>Read/Write</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>Read/Write</td>
<td>Read-only</td>
<td>No</td>
</tr>
<tr>
<td>Read/Write</td>
<td>Read/Write</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Prefetch processing
DB2 prefetch processing for GBP-dependent page sets and partitions varies depending on the level of coupling facility (CFLEVEL) in which the group buffer pool is allocated.

If the group buffer pool is allocated in a coupling facility with CFLEVEL=0 or 1, DB2 reads and registers one page at a time in the group buffer pool.

If the group buffer pool is allocated in a coupling facility with CFLEVEL=2 or higher, DB2 can register the entire list of pages that are being prefetched with one request to the coupling facility. This can be used for sequential prefetch (including sequential detection) and list prefetch. On the list, DB2 does not include any valid pages that are found in the local buffer pool.

For those pages that are cached as “changed” in the group buffer pool, or those that are locked for castout, DB2 still retrieves the changed page from the group buffer pool one at a time. For large, sequential queries, changed pages most likely do not exist in the group buffer pool.

For pages that are cached as “clean” in the group buffer pool, DB2 can get the pages from the group buffer pool (one page at a time), or can include the pages in the disk read I/O request, depending on which is most efficient.

Determining if DB2 registered a list of pages: If DB2 registers a list of pages during prefetch, there will be a non-zero value in field QBGLAX in IFCID 0002 (see Figure 70 on page 267). You can also use the DISPLAY GROUPBUFFERPOOL command with the MDETAIL option (see message DSNB789J).

Caching pages that are read in from disk
You can cache pages in the group buffer pool as they are read into a member’s local buffer pool by specifying GBPCACHE ALL when you create or alter a table space or index. When you choose ALL, pages are copied to the group buffer pool as they are read in from disk, even if no inter-DB2 read/write interest exists in those pages.

However, when only a single member has exclusive read/write interest in the page set (that is, only one member has the page set open for update), pages are not cached in the group buffer pool when they are read in from disk. As soon as another member in the data sharing group shows interest in the page set or partition, it becomes GBP-dependent. Changed pages are moved into the group buffer pool, and all pages read in from disk are cached in the group buffer pool.

Choosing GBPCACHE ALL does not prevent DB2 from continuing to cache changed pages in the group buffer pool before writing them to disk (the function provided by the default, GBPCACHE CHANGED).

General-use Programming Interface

Example: Use the GBPCACHE clause to cache read-only page sets in the group buffer pool:

```
ALTER TABLESPACE DSNB081A.DSNB81D
  GBPCACHE ALL;
```
See Chapter 5 of DB2 SQL Reference for more information about the GBPCACHE clause.

End of General-use Programming Interface

Why choose GBPCACHE ALL?: By choosing GBPCACHE ALL, you prevent multiple members from reading the same page in from disk. For this reason, page sets or partitions that typically have a high degree of inter-DB2 read interest are good candidates for GBPCACHE ALL. (To prevent the double buffering of clean pages, hiperpools are not used for page sets or partitions that are defined with GBPCACHE ALL.)

Planning consideration: If you use the GBPCACHE ALL option, it increases the need for coupling facility resources: processing power, storage, and channel utilization. If you cannot afford the additional strain on coupling facility resources, consider using a 3990 Model 6 cache controller that exploits record and track level caching to achieve caching benefits for a read-intensive work load.

Recommendation: For LOB table spaces, use GBPCACHE SYSTEM, which is the default for LOB table spaces. If you use GBPCACHE ALL or GBPCACHE CHANGED for a LOB table space that is defined with LOG NO and the coupling facility fails, the LOB table space is placed in GRECP. When group buffer pool recovery occurs, all LOB values that were in the coupling facility at the time of the failure are marked invalid because the log records that are necessary to perform the recovery for those values are missing due to the LOG NO attribute.

Write operations
With data sharing, DB2 usually writes changed data to the group buffer pool before writing that changed data to disk.

This section includes the following topics:
• “How the GBPCACHE option affects write operations”
• “Writing to the group buffer pool” on page 250
• “Writing to disk from the group buffer pool” on page 251

How the GBPCACHE option affects write operations
You can use several options in a CREATE TABLESPACE statement to tell DB2 how you want data to be handled through the group buffer pool. This section describes the various options and in what cases each might be appropriate.

GBPCACHE CHANGED or ALL: Most of the time you use GBPCACHE CHANGED. (The discussion of when to use GBPCACHE ALL is in “Caching pages that are read in from disk” on page 247.) For both ALL and CHANGED, the write operations are the same for changed pages: changed pages are written to the group buffer pool before being written to disk. Details about write operations are described in “Writing to the group buffer pool” on page 250.

GBPCACHE SYSTEM: This option is allowed only for LOBs. For GBPCACHE SYSTEM page sets, the only pages that are written to the group buffer pool are LOB space map pages. All other data pages are written directly to disk, similar to GBPCACHE NONE page sets. GBPCACHE SYSTEM is the default for a LOB table space.

Recommendation: In a data sharing environment, if GBPCACHE CHANGED or GBPCACHE ALL is used instead for a LOG NO LOB table space, a coupling facility failure can result in the table space being marked GRECP. Any kind of
recovery of the table space marks the attached LOB values as invalid and places the table space in the AUXW state. For LOB table spaces, choose GBPCACHE SYSTEM to avoid having large LOB values overwhelm the group buffer pool. Also, for LOB table spaces with the LOG NO attribute, GBPCACHE SYSTEM ensures that LOB values are written to disk by commit time, thereby avoiding possible recovery problems caused by missing log data.

**GBPCACHE NONE:** For GBPCACHE NONE page sets, or for page sets defined in a group buffer pool as GBPCACHE(NO), no pages are written to the group buffer pool. The group buffer pool is used only for the purpose of buffer cross-invalidation. At every COMMIT, any pages that were updated by the transaction and have not yet been written are synchronously written to disk during commit processing. This can have a severe impact on performance for most types of transactions.

One potential advantage of not caching pages in the group buffer pool is that data does not have to be recovered from the log if the coupling facility fails. However, because DB2 still depends on the cross-invalidation information that is stored in the group buffer pool, a coupling facility failure still means some data might not be available. That is why specifying an alternate coupling facility in the CFRM policy is recommended. If you are looking for a high availability option, consider duplexing the group buffer pool rather than suffering the performance hit of writing directly to disk at every COMMIT.

**Advantage of specifying GBPCACHE(NO) for group buffer pools:** You can specify GBPCACHE(NO) on the group buffer pool level instead of on the page set level. Changing the group buffer pool attribute is usually less disruptive than changing the page set attribute. Because the GBPCACHE(NO) attribute takes precedence over the GBPCACHE option on the page set, you can plan to use different types of processing at different times of day. For example, assume that you want to run transactions or queries during the day, but you want to do batch updates at night. Assume also that your batch updates would benefit from no group buffer pool data caching. You can do the following:

1. Define the table space as GBPCACHE CHANGED and put it into its own buffer pool.
2. Define the corresponding group buffer pool as GBPCACHE(YES) for daytime processing.
3. At night, use the ALTER GROUPBUFFERPOOL command to change the group buffer pool to GBPCACHE(NO).
4. Set the SETXCF START,REBUILD,STRNAME=strname command to enable the new attribute.
5. In the morning, use the ALTER GROUPBUFFERPOOL command to change the group buffer pool back to GBPCACHE(YES).
6. Issue the SETXCF START,REBUILD,STRNAME=strname command to enable the new attribute.

**Reasons not to cache:** A performance benefit is possible for applications in which an updated page is rarely, if ever, referenced again, such as a batch job that sequentially updates a large table. By not caching, you save the costs of transferring data to the group buffer pool and casting out from the group buffer pool. Again, this benefit is at the cost of synchronous disk I/O at COMMIT. To reduce the amount of synchronous disk I/O at COMMIT, you can lower the deferred write thresholds so that DB2 writes more pages asynchronously prior to COMMIT.
To determine if a particular group buffer pool is a candidate for GBPCACHE(NO), look at the group buffer pool statistics. If the ratio of READS, DATA RETURNED / PAGES WRITTEN is less than 1%, this page set might be a good candidate for GBPCACHE(NO), or the page sets using this group buffer pool might be a good candidate for GBPCACHE NONE. You receive the following benefits:

- Reduced coupling facility costs and faster coupling facility response time
- Reduced processor time on the host system
- Better transaction throughput at a small possible cost in higher transaction response time

If you use GBPCACHE NONE, enable DASD Fast Write and set the vertical deferred write threshold (VDWQT) to 0. By setting this threshold to 0, you let deferred writes happen continuously before the COMMIT, thus avoiding a large surge of write activity at the COMMIT.

**Writing to the group buffer pool**

With data sharing, DB2 still performs deferred writes for DB2 table spaces, indexes or partitions. However, when an update is to a page set that has inter-DB2 read/write interest, DB2 forces the updated pages to the group buffer pool before or when the transaction commits. Updated pages can be written to the group buffer pool before the updating transaction is committed when:

- One of the deferred write thresholds is reached.
- The buffer pool lacks reassignable buffers because writes to the group buffer pool cannot keep up with update activity in the buffer pool. The shortage of buffers can occur when the deferred write thresholds are too high, or if the application is not committing frequently enough—in a data sharing environment, the commits make buffers reassignable.
- An updated page has stayed in the buffer pool for a long period of time since it was last referenced or updated (such as with a long-running transaction that does not issue frequent commits). In this case, a system checkpoint can free space in the buffer pool before the commit.
- The same page is required for update by another system because no conflict on transaction locking exists (such as page sets that are using row locking, index pages, space map pages, and so on). This write is part of the page P-lock negotiation process.

When a page of data is written to the group buffer pool, all copies of that page cached in other members’ buffer pools are invalidated. This means that the next time that one of those members needs that page, the page must be refreshed from the group buffer pool (or disk).

Before an updated page is written to the group buffer pool or to disk, DB2 also ensures that the last update log record for that page is externalized to the active log. This ensures that updates can be backed out when necessary.

When committing an updating transaction, DB2 synchronously writes to the group buffer pool pages that were updated but not yet written to the group buffer pool. If a group buffer pool is required and unavailable (because of a channel or hardware failure) at the time the transaction commits, DB2 places all the transaction’s updated pages on the logical page list (LPL) associated with each page set. After the problem is fixed, DB2 attempts automatic LPL recovery, but you can issue a START DATABASE command with the SPACENAM option to manually recover the pages on the logical page list. Refer to [“Monitoring databases” on page 154](#) for more information about LPL recovery.
**Writing to a GBPCACHE(NO) group buffer pool:** For GBP-dependent page sets, no data is written to a group buffer pool for the following instances:

- The group buffer pool is defined as GBPCACHE(NO)
- The page set is defined as GBPCACHE NONE
- The changed pages are non-system pages of a page set defined with GBPCACHE SYSTEM

In these cases, DB2 writes changed pages for the transaction directly to disk at or before COMMIT. DB2 batches up to 32 pages in a single I/O.

When DB2 writes a changed page to disk, it cross-invalidates the page using the group buffer pool. These cross-invalidations are called explicit cross-invalidations. These explicit cross-invalidations are reported in statistics separately from cross-invalidations caused by directory reclaims or by writes of a changed page to a group buffer pool.

**Writing to a duplexed group buffer pool:** When a group buffer pool is duplexed, page writes are performed in the following manner:

1. For some fixed number of pages that must be written:
   a. Each page is written to the secondary group buffer pool asynchronously
   b. Each page is written to the primary group buffer pool synchronously
2. After all pages have been written to the primary group buffer pool, DB2 checks to see if all pages have been written to the secondary group buffer pool. If some pages still need to be written, DB2 forces the completion of those writes.

**Writing to disk from the group buffer pool**

The process of writing pages from the group buffer pool to disk is called **castout**. Because no physical connection exists between the group buffer pool and disk, the castout process involves reading the page from the group buffer pool into a particular member’s private buffer (not part of the buffer pool storage), and then writing the page from the private buffer to disk. This member is the owner of the castout process for the page set or partition. The first member with update intent on the page set or partition is assigned ownership of castout. After castout ownership is assigned, subsequent updating members become backup owners. One of the backup owners becomes the castout owner when the original castout owner no longer has read/write interest in the page set.

Other members can write this page to the group buffer pool even as the page is being cast out. Some events explicitly cause pages to be cast out to disk, such as the STOP DATABASE command.

Castout also occurs when:

- The number of changed pages for a castout class **queue exceeds a class threshold value**. Castout class thresholds are described in “**Group buffer pool class castout threshold**” on page 256.
- The total number of changed pages for a **group buffer pool exceeds a group buffer pool threshold value**, described in “**Group buffer pool castout threshold**” on page 256.
- The **group buffer pool checkpoint** is triggered. See “**Group buffer pool checkpoint**” on page 253 for more information.
- No more inter-DB2 read/write interest exists in the page set.
- The **group buffer pool is being rebuilt**, but the alternate group buffer pool is not large enough to contain the pages from the group buffer that is being rebuilt.
Pages that are cast out as a result of meeting a threshold remain cached in the group buffer pool, and the buffers are available for stealing. Pages that are cast out because no more shared interest exists in the page set are purged from the group buffer pool.

**Casting out from a duplexed group buffer pool:** DB2 casts out data to disk only from the primary structure. After a set of pages has been cast out, the same set of pages is deleted from the secondary structure. See the DELETE NAME LIST counter in the DISPLAY GROUP BUFFERPOOL MDETAIL report for how many times this event occurs. DB2 ensures that any pages that might have been written to the group buffer pool during castout processing are not deleted from the secondary structure.

**Displaying the castout owner:**

---

**General-use Programming Interface**

---

Use the DISPLAY DATABASE command with the LOCKS option to display the current castout owner for a given page set:

```
-DB1A DISPLAY DATABASE(TESTDB) SPACE(*) LOCKS
```

Display the castout owner for a particular page set or partition with (CO) by the member name, as shown in Figure 59.

```
: TBS43 TS 01 RW MEMBER NAME DB2A (CO) H-SIX,PP,I
- TBS43 TS 02 RW MEMBER NAME DB1A BATCH SELEC H-IS,P,C
: : :
```

*Figure 59. Partial DISPLAY DATABASE output showing castout owner for a partition*

---

**End of General-use Programming Interface**

---

Use the DISPLAY BUFFERPOOL command with the CASTOWNR keyword to determine for which page sets or partitions the members hold castout ownership:

```
-DIS BUFFERPOOL(BP0) CASTOWNR(Y)
```

*Figure 60 on page 253* shows output that lists the members that hold the page set or partition P-lock in "U" state.
GROUPBUFFERPOOL

You structure the owner.
The take record buffer reduce primary the group.
When Group page Figure DSN9022I

DATABASE

-------------PAGE

DSNB460I

DSNB404I

DSNB402I

DSNB401I

@DIS

Partial DISPLAY DATABASE output showing the list of members that hold the page set or partition P-lock in "U" state

Figure 60.

Group buffer pool checkpoint

When a group buffer pool is damaged, all changed data that belong to GBP-dependent page sets must be recovered to the page sets from the DB2 logs. The number of log records that need to be applied to the page set is determined by the frequency of the group buffer pool checkpoint. Group buffer pool checkpoint is the process of writing all changed pages in the group buffer pool (only the primary one, if duplexed) to the page set. The purpose of the checkpoint is to reduce the amount of time needed to recover data in a group buffer pool. At group buffer pool checkpoint, DB2 records, in the member BSDSs and SCA, the log record sequence number from which group buffer pool recovery would need to take place. Group buffer pool checkpoint does not record anything in the log.

The group buffer pool checkpoint is triggered by the structure owner. The structure owner is usually the first member that connects to this group buffer pool, although the ownership can change over time. Message DSNB798I in the DISPLAY GROUPBUFFERPOOL command output shows which member is the current structure owner.

Default checkpoint frequency: The default checkpoint frequency is 8 minutes. You can change the default checkpoint frequency by using the ALTER GROUPBUFFERPOOL command, described in "Changing the checkpoint frequency” on page 270. For group buffer pools defined as GBPCACHE(NO), the checkpoint interval is ignored; no checkpointing occurs for those group buffer pools.

Tuning the group buffer pool checkpoint interval: At the group buffer pool checkpoint, the structure owner records, in the SCA and in its own BSDS, the
LRSN from which group buffer pool recovery should take place, if necessary. This LRSN is displayed in the DISPLAY GROUPBUFFERPOOL command output.

DB2 has two possible ways of gathering checkpoint information:

• By issuing many “read directory info” requests
  This method is used when the Parallel Sysplex is not at the maintenance level required for the more efficient method described below. These “read directory info” requests are reported as an increase in the SRB time of the ssnm=DBM1 address space, especially for the structure owner. This is because of the increased number of times that DB2 has to read the directory entries to compute the recovery LRSN.

• By issuing one “read castout statistics” request
  This method is used when the group buffer pool is allocated in a coupling facility at CFLEVEL=5 or higher.
  When you look at the MDETAIL report from a DISPLAY GROUPBUFFERPOOL command, as shown in Figure 62 on page 255, you will see significantly fewer “read directory info” requests when you have the proper maintenance applied for this feature.

**Recommendation:** Because group buffer pool checkpoint consumes processor, coupling facility, and I/O resources and can impact other work in the system, balance the performance impact of frequent group buffer pool checkpoints (the lower the checkpoint interval, the higher the system resource consumption) with the recovery impact of infrequent checkpoints (the lower the checkpoint interval, the faster DB2 can recover from a group buffer pool failure). The default checkpoint interval of 8 minutes is a good balance between the performance and recovery considerations in most cases.

If the resource consumption of the group buffer pool checkpoint is higher than you prefer:

• Apply the proper maintenance and allocate the group buffer pool in a CFLEVEL=5 coupling facility to take advantage of the checkpoint performance enhancement.

• Increase the checkpoint interval to have the checkpoint occur less frequently.

If the checkpoint is not moving the recovery LRSN forward fast enough, decrease the checkpoint interval. You can determine the LRSN by periodically issuing the DISPLAY GROUPBUFFERPOOL command.

---

**General-use Programming Interface**

The following instrumentation helps you more effectively monitor and tune the group buffer pool checkpoint:

• The DISPLAY GROUPBUFFERPOOL command in Figure 61 shows which member is the structure owner and also shows the group buffer pool checkpoint recovery LRSN.

**Figure 61. Partial DISPLAY GROUPBUFFERPOOL output showing which member owns the structure and the group buffer pool checkpoint recovery LRSN**
• The DISPLAY GROUPBUFFERPOOL command with MDETAIL option in Figure 62 contains the number of checkpoints that occurred for this group buffer pool. The statistics trace also includes this information.

DSNB778I -DB1A CASTOUT THRESHOLDS DETECTED
FOR CLASSES = 3
FOR GROUP BUFFER POOL = 1
GBP CHECKPOINTS Triggered = 1
PARTICIPATION IN REBUILD = 0

DSNB796I -DB1A CASTOUTS
PAGES CASTOUT = 18
UNLOCK CASTOUT = 3
READ CASTOUT CLASS = 5
READ CASTOUT STATISTICS = 6
READ DIRECTORY INFO = 0

Figure 62. DISPLAY GROUPBUFFERPOOL MDETAIL report

If you are experiencing surges of coupling facility use, it could be related to group buffer pool checkpointing. Examine the number of read directory info requests. If there are many requests per checkpoint, you can probably benefit from applying the proper maintenance to use the group buffer pool checkpoint performance enhancement.

• IFCID 0261, which gives summary statistics for each group buffer pool checkpoint. You can use this record to estimate the processor cost and to monitor the coupling facility interactions for each group buffer pool checkpoint.

• IFCID 0263, which gives summary statistics for the castouts. You can use this record to monitor the castout activity that is caused by each group buffer pool checkpoint (or triggered for any other reason).

End of General-use Programming Interface

If the recovery LRSN for group buffer pool checkpoint is not advancing as fast as you want, determine if there have been any disk or coupling facility connectivity problems that are impairing the ability of DB2 to cast out.

Group buffer pool thresholds

You can control the castout process by changing the following two group buffer pool thresholds:
• Group buffer pool castout threshold
• Class castout threshold

These thresholds have no effect for GBPCACHE(NO) group buffer pools.

As Figure 63 on page 256 illustrates, the group buffer pool castout threshold is a percentage of changed pages in the group buffer pool. The class castout threshold is the percentage of changed pages in the group buffer pool per castout queue.
Group buffer pool class castout threshold

In each group buffer pool there is a fixed number of castout class queues. This number is an internal value set by DB2. DB2 internally maps updated pages that belong to the same page sets or partitions to the same castout class queues. Because of a limited number of castout class queues, it is possible that more than one page set or partition gets mapped into the same castout class queue. This internal mapping scheme is the same across all sharing subsystems.

When DB2 writes changed pages to the group buffer pool, it determines how many changed pages are on a particular class castout queue. If the number of changed pages on a specified castout class queue exceeds the threshold, DB2 casts out a number of pages from that queue.

How DB2 determines the castout threshold for a duplexed group buffer pool: For duplexed group buffer pools, DB2 uses the smaller of the number of data entries in the primary and secondary structures. For example, if the primary structure contains 5000 data entries and the secondary structure contains 1000 data entries, and CLASST is 10%, DB2 sets CLASST to 100 pages (10% of 1000 pages).

Default group buffer pool class castout threshold: The default for the class castout is 10, which means that castout is initiated for a particular page set or partition when 10% of the group buffer pool contains changed pages for the class.

Group buffer pool castout threshold

This threshold determines the total number of changed pages that can exist in the group buffer pool before castout occurs. DB2 casts out enough class castout queues to bring the number of changed pages below the threshold. DB2 periodically determines whether the threshold is exceeded.
How DB2 determines the group buffer pool castout threshold for a duplexed group buffer pool: For duplexed group buffer pools, DB2 uses the smaller of the number of data entries in the primary and secondary group buffer pools. For example, if the primary structure contains 5000 data entries and the secondary structure contains 1000 data entries, and GBPOOLT is 50%, DB2 sets GBPOOLT to 500 pages (50% of 1000 pages).

Default group buffer pool castout threshold: The default value for the group buffer pool castout threshold is 50, which means that when the group buffer pool is 50% full of changed pages, castout is initiated.

Guidelines
In most cases, you should use the CLASST and GBPOOLT thresholds to be the corresponding VDWQT and DWQT thresholds for the local buffer pools if these values perform well locally. Otherwise, you can use the default values (10% for the class threshold and 50% for the group buffer pool threshold). Depending on your work load, these values help reduce disk contention during castout.

If you find that some writes to the group buffer pool cannot occur because of a lack of storage in the group buffer pool, increase the group buffer pool size, or decrease the group buffer pool castout thresholds. One way to tell if this is happening is to see the detail report of the DISPLAY GROUPBUFFERPOOL command. An example report is shown in Figure 68 on page 264. The field indicated by [E] indicates this type of problem.

Tuning the castout thresholds: The following facilities can help you more effectively monitor the group buffer pool castout thresholds:

- The DISPLAY GROUPBUFFERPOOL command with the MDETAIL option.
- The DB2 statistics trace.
- IFCID 0262, which gives summary statistics for each time that the GBPOOLT threshold is reached. You can use this record to monitor how efficiently the GBPOOLT threshold is handling the castout work.
- IFCID 0263, which gives summary statistics for the castouts done by the page set and partition castout owners. All castout work for a given page set or partition is done by the castout owner. You can use this record to monitor the efficiency with which the page set or partition castout owners are doing their work.

Example from MDETAIL report:

<table>
<thead>
<tr>
<th>General-use Programming Interface</th>
</tr>
</thead>
</table>

Here is partial output from the command DISPLAY GROUPBUFFERPOOL (GBP0) MDETAIL:

```
DSNB796I -DB1A CASTOUTS

PAGES CASTOUT = 217
UNLOCK CASTOUT = 35
READ CASTOUT CLASS = 47
READ CASTOUT STATISTICS = 47
READ DIRECTORY INFO = 290
```

The UNLOCK CASTOUT counter should always be significantly less than the PAGES CASTOUT counter. If, at the very least, it is not less than half, the castout write I/O is not performed efficiently. (The number of pages written per I/O is normally close to the number that is obtained by dividing PAGES CASTOUT by
UNLOCK CASTOUT). This is probably because you have random update patterns on the DB2 data.

_________________________
End of General-use Programming Interface

Effect of GBPCACHE ALL on guidelines: If you are using a group buffer pool to cache pages as they are read in from disk (GBPCACHE ALL page sets), consider lowering the threshold values to allow more space for caching those clean pages.

Monitoring group buffer pools
This section describes how you can monitor group buffer pool activity:
- "Using the z/OS DISPLAY XCF,STR command"
- "Using the coupling facility activity report of RMF" on page 259
- "Using the DISPLAY GROUPBUFFERPOOL command" on page 260
- "Using DB2 statistics trace" on page 262

Using the z/OS DISPLAY XCF,STR command
You can use z/OS command D XCF,STR to get information about coupling facility structures:
- CFRM policy definition
- Preference list
- Coupling facility name
- Connections
- Duplexing status

The following command displays information about GBP1 in group DSNDB0A:
D XCF,STR,STRNAME=DSNDB0A_GBP1

This particular group buffer pool is duplexed, so you see information about both allocations of the structure (the old structure is the primary structure, and the new structure is the secondary one). The output is similar to the output that is shown in Figure 64 on page 259.
DISPLAY XCF
STRNAME: DSNDB0A GBP1
STATUS: REASON SPECIFIED WITH REBUILD START:
        OPERATOR INITIATED

DUPLLEXING REBUILD
REBUILD PHASE: DUPLEX ESTABLISHED
POLICY SIZE : 204800 K
POLICY INITSIZE: 102400 K
REBUILD PERCENT: 1
DUPLEX : ALLOWED
PREFERENCE LIST: CACHE01, LF01
EXCLUSION LIST IS EMPTY

DUPLLEXING REBUILD NEW STRUCTURE
-------------------------------
ALLOCATION TIME: 10/14/2002 17:01:48
CFNAME : LF01
COUPLING FACILITY: ND01...
        PARTITION: 0  CPCID: 00
ACTUAL SIZE : 102400 K
STORAGE INCREMENT SIZE: 256 K
VERSION : AF6935AA 70004403
DISPOSITION : DELETE
ACCESS TIME : 0
MAX CONNECTIONS: 32
# CONNECTIONS : 2

DUPLLEXING REBUILD OLD STRUCTURE
-------------------------------
ALLOCATION TIME: 10/14/2002 17:00:38
CFNAME : CACHE01
COUPLING FACILITY: ND02...
        PARTITION: 0  CPCID: 00
ACTUAL SIZE : 102400 K
STORAGE INCREMENT SIZE: 256 K
VERSION : AF693567 9B48B802
ACCESS TIME : 0
MAX CONNECTIONS: 32
# CONNECTIONS : 2

CONNECTION NAME  ID  VERSION  SYSTYPE  JOBNM  ASID  STATE
------------------  ---  -------  ------  -----  ----  ----
DB2_DB1A  01  00010001  UTEC469  DB1ADBM1  002E  ACTIVE  NEW,OLD
DB2_DB2A  02  00020001  UTEC469  DB2ADBM1  0031  ACTIVE  NEW,OLD

Figure 64. z/OS Command D XCF showing group buffer pool information

For more information about the D XCF command, see z/OS MVS System Commands.

Using the coupling facility activity report of RMF
See the portion of an RMF coupling facility structure report show in Figure 65 on page 260. A value for CHNGD (B) is the percentage of all accesses that were supposed to be done synchronously that had to be done asynchronously. The NO SCH field (A) indicates the amount of time that requests were queued because of a lack of subchannel resources. If the value in B is over 10% or so, and A contains a non-zero value, your configuration might not have enough subchannels to handle your systems’ workload.
Using the DISPLAY GROUPBUFFERPOOL command

Use the DISPLAY GROUPBUFFERPOOL command to display information about group buffer pools. Assume that you want a summary report about group buffer pool zero (GBP0), including all connections to that group buffer pool. Enter the following command:

-DB1A DISPLAY GROUPBUFFERPOOL(GBP0) CONNLIST(YES)

Figure 66 on page 261 shows what the display might look like, assuming that the group buffer pool is duplexed.
See Part 3 of [DB2 Command Reference](#) for more information about the syntax of the command.

You can display detailed statistics using the GDETAIL and MDETAIL keywords. See [Figure 68 on page 264](#) to see what statistical information looks like.

**Monitoring delete name requests:** Use the DISPLAY GROUPBUFFERPOOL command with the MDETAIL option to determine how many times delete-name requests occur. [Figure 67 on page 262](#) shows partial output of the DISPLAY GROUPBUFFERPOOL command which includes delete-name information:

- DISPLAY GROUPBUFFERPOOL(GBP29) MDETAIL(*)
You can also monitor delete-name requests by using the OMEGAMON Statistics Detail report or IFCID 263 (Page set castout detail). See Figure 70 on page 267 for an example of a OMEGAMON Statistics Detail report, which contains delete-name information.

Recommendation: For an easy way to collect interval statistics for performance analysis, create a batch job that issues the following command periodically:

```
-DB1A DISPLAY GROUPBUFFERPOOL(+) GDETAIL(INTERVAL)
```

The first time you run the batch job is the base which purges existing statistics and resets the interval. If you run the job the second time 5 minutes after the first, you can continue running the job every 5 minutes to gather meaningful statistical data on group buffer pool activity.

Using DB2 statistics trace

Use DB2 statistics class 1 to do high-level monitoring of DB2 subsystem activity. A data section mapped by DSNDQBGL records statistics for a member’s use of group buffer pools. The counters are cumulative since the time the member connected to a particular group buffer pool.

Statistics reporting intervals are not synchronized across the members of the data sharing group. For counters that are pertinent to the entire data sharing group, like group buffer statistics, OMEGAMON group-scope statistics reports combine the data of the individual members and present it for the entire group. The member data is apportioned to the same user-specified interval. OMEGAMON presents the synchronized statistics intervals for each member, adds the counters across all members and presents them as statistics on a per-group basis.

Consider also using OMEGAMON to do group-scope exception reporting when a particular counter exceeds a user-specified value.

For more information about using the statistics report, see “What to look for in a OMEGAMON statistics report” on page 267.
Determining the correct size and ratio

One of the critical tuning factors in a DB2 data sharing configuration is the size of the group buffer pools. Three aspects of group buffer pool (cache structure) size need to be considered:

- **Total structure size**
  
  As described in "General information about coupling facility storage" on page 47, the total structure size of a group buffer pool is specified in the coupling facility policy definition for the cache structure.

- **Number of directory entries**

  A directory entry is used by the coupling facility to determine where to send cross-invalidation signals when a page of data is changed or when that directory entry must be reused. A directory entry contains control information for one database page, no matter in how many places that page is cached. For example, if page P1 is cached in the group buffer pool and in the buffer pools of three members, that page still has only one directory entry.

  See IBM eServer zSeries Processor Resource/System Manager Planning Guide for information about the size of directory entries for your CFLEVEL

- **Number of data entries**

  Data entries are the actual places where the data page resides. These are 4-KB, 8-KB, 16-KB, or 32-KB in size (the same size as the data page).

  For GBPCACHE NO group buffer pools, no data entries exists.

  The number of directory entries and data entries in the coupling facility structure is determined by the size specified in the coupling facility policy and the ratio of directory entries to data pages. The ratio is automatically defined for each group buffer pool at the time that the first member of the group is installed. The default value used is 5 directory entries per data page.

  For secondary group buffer pools, the ratio is the same as the ratio used for the primary group buffer pools.

  For formulas to help you choose a ratio, see "Group buffer pool sizes" on page 48.

  After installation, you can change the ratio with the ALTER GROUPBUFFERPOOL command. However, the change does not take effect until the next time the group buffer pool is allocated.

  The following sections describe the symptoms and values that are not ideal for best performance, and discuss how you can fix the problems.

**Group buffer pool size is too small**

When the group buffer pool is too small, the following problems can occur:

- The thresholds for changed pages are reached more frequently, causing data to be cast out to disk more often.

  If castout cannot keep up with the writes to the group buffer pool, a more serious problem occurs: pages are instead written to the logical page list and are unavailable until they are recovered. See "Monitor storage of the group buffer pool" on page 265 for a recommendation about avoiding this problem. See "Problem: storage shortage in the group buffer pool" on page 178 for recovery actions should the problem occur.

- Many cross-invalidations caused by reusing existing directory entries, which might require refreshing a page from disk later when the page is referenced again.
Pages in the group buffer pool need to be refreshed from disk more often because they are not in the group buffer pool. You can use the GDETAIL option of the DISPLAY GROUPBUFFERPOOL command to gather detailed statistical information about how often data is returned on a read request to the group buffer pool:

```
-DB1A DISPLAY GROUPBUFFERPOOL(GBP0) GDETAIL(*)
```

Figure 68 shows output similar to the detail portion of the report output.

```
DSNB783I -DB1A CUMULATIVE GROUP DETAIL STATISTICS SINCE 15:35:23 May 17, 2002
DSNB784I -DB1A GROUP DETAIL STATISTICS
READS
   DATA RETURNED A = 3845
   DATA NOT RETURNED
      DIRECTORY ENTRY EXISTED B = 27
      DIRECTORY ENTRY CREATED C = 28336
      DIRECTORY ENTRY NOT CREATED D = 332,000
DSNB785I -DB1A WRITES
   CHANGED PAGES = 20909
   CLEAN PAGES = 0
   FAILED DUE TO LACK OF STORAGE E = 8
   CHANGED PAGES SNAPSHOT VALUE = 974
DSNB786I -DB1A RECLAIMS
   FOR DIRECTORY ENTRIES F = 18281
   FOR DATA ENTRIES = 47
   CASTOUTS = 16073
DSNB787I -DB1A CROSS INVALIDATIONS
   DUE TO DIRECTORIES RECLAIMS G = 4489
   DUE TO WRITES = 3624
   EXPLICIT = 0
DSNB788I -DB1A DUPLEXING STATISTICS FOR GBP0-SEC
   CHANGED PAGES = 20909
   FAILED DUE TO LACK OF STORAGE = 8
   CHANGED PAGES SNAPSHOT VALUE = 974
DSNB790I -DB1A DISPLAY FOR GROUP BUFFER POOL GBP0 IS COMPLETE
DSNB922I -DB1A DSNB1CMD 'DISPLAY GROUPBUFFERPOOL' NORMAL COMPLETION
```

**Figure 68. Example output of group detail statistics**

What you need to determine is the read-hit percentage. To calculate this value, you need to determine how many of the total number of reads were successful in returning data. Use the following formula:

\[
\text{read-hit percentage} = \left( \frac{A}{A + B + C + D(\text{first number})} \right) \times 100
\]

In this example, the calculation is:

\[
(3845 / 32540) \times 100 = 11.81\%
\]

Data was returned in approximately 12% of the read requests to the group buffer pool. This low percentage of read hits might indicate that the average residency time for a cached page in group buffer pool is too short. You might benefit from altering the group buffer pool to increase the total size, as described in “Changing the size of the group buffer pool” on page 270.

However, a low percentage of read hits could be caused by other factors:

- A high read-to-write ratio.
If you are caching only changed pages, not many pages you need will be resident in the group buffer pool.

- Random reference patterns.

Pages that are frequently referenced are most likely to be resident in the group buffer pool. If the application keeps requesting new pages, any given page is unlikely to be found in the group buffer pool.

To determine whether the low read-hit percentage is a problem, see the field indicated by \[ \text{A} \] in the statistics report, shown in Figure 70 on page 267 (The same counter also exists in the accounting report.) Ideally, that field contains \( 0 \). A non-zero value in the field, in conjunction with a low read hit percentage, can indicate that your group buffer pool is too small.

“Too few directory entries” on page 266 and “Too few data entries” on page 266 describe how to determine if the problem is caused by a suboptimal ratio of directory entries to data entries.

---

End of General-use Programming Interface

Monitor storage of the group buffer pool:

---

General-use Programming Interface

By monitoring the storage use of the group buffer pool, you can avoid data outages caused by a serious lack of storage in the group buffer pool.

Recommendation: Issue periodic DISPLAY GROUPBUFFERPOOL commands with the GDETAIL option. The GDETAIL statistics show a snapshot value of the number of changed pages in the group buffer pool. Ensure that this snapshot value (\[ \text{A} \]) does not rise significantly above the group buffer pool castout threshold (\[ \text{B} \times \text{C} \]). Figure 69 highlights the key fields from the report.

```
DSNB756I -DB1A CLASS CASTOUT THRESHOLD = 10%
  A GROUP BUFFER POOL CASTOUT THRESHOLD = 50%
  GROUP BUFFER POOL CHECKPOINT INTERVAL = 8 MINUTES
  RECOVERY STATUS = NORMAL
  AUTOMATIC RECOVERY = Y
DSNB759I -DB1A NUMBER OF DIRECTORY ENTRIES = 61394
  B NUMBER OF DATA PAGES = 11370
  NUMBER OF CONNECTIONS = 3
DSNB783I -DB1A CUMULATIVE GROUP DETAIL STATISTICS SINCE 15:35:23 Mar 17, 2002
DSNB784I -DB1A GROUP DETAIL STATISTICS
  
DSNB786I -DB1A WRITES
  CHANGED PAGES = 1576
  CLEAN PAGES = 0
  FAILED DUE TO LACK OF STORAGE = 0
  CHANGED PAGES SNAPSHOT VALUE = 311
  
```

Figure 69. Partial output of DISPLAY GROUPBUFFERPOOL command. Ensure that the SNAPSHOT value does not rise significantly above the group buffer pool castout threshold.

---

End of General-use Programming Interface
Too few directory entries

When existing directory entries are being reclaimed to handle new work, cross-invalidation must occur for all of the members that have the particular data pages in their buffer pools, even when the data has not actually changed.

For example, in Figure 68 on page 264, E indicates that there have been 18,281 directory reclaims. C indicates that, because of those reclaims, 4,489 cross-invalidations occurred. The pages in those members' buffer pools need to be refreshed when next needed, probably from disk, which can degrade system performance.

If there is a high value in C, check the group buffer pool hit percentage (described in “Group buffer pool size is too small” on page 263) to see if the lack of directory entries might be causing an excessive number of reads from disk.

You can also check the “Synchronous Reads Due To Buffer Invalidations” counters shown in the member detail report.

You can increase the number of directory entries in the group buffer pool in one of two ways:

- Increase the total size of the group buffer pool, as described in “Changing the size of the group buffer pool” on page 270.
- Use the ALTER GROUPBUFFERPOOL command to adjust the ratio in favor of directory entries, as described in “Changing the ratio of directory to data entries” on page 272.

Too few data entries

If a group buffer pool does not have enough data entries, castout to disk occurs more frequently. You can see the number of pages cast out by using the GDETAIL option of the DISPLAY GROUPBUFFERPOOL command.

A more serious data entry shortage is indicated by field E in the DISPLAY GROUPBUFFERPOOL GDETAIL report shown in Figure 68 on page 264. A value in this field indicates that the data page resources of the coupling facility are being consumed faster than the DB2 castout processes can free them.

You can increase the number of data entries in the group buffer pool in one of two ways:

- Increase the total size of the group buffer pool, as described in “Changing the size of the group buffer pool” on page 270.
- Use the ALTER GROUPBUFFERPOOL command to adjust the ratio in favor of data entries, as described in “Changing the ratio of directory to data entries” on page 272.
What to look for in a OMEGAMON statistics report

Refer to the OMEGAMON statistics detail report shown in Figure 70. The fields from that report are used to explain some of the activity that takes place for cross-invalidation and refresh of buffers. Some of the information is the same as that described by the DISPLAY GROUPBUFFERPOOL output, which was covered earlier in this chapter.

<table>
<thead>
<tr>
<th>GROUP BP4</th>
<th>QUANTITY /MINUTE</th>
<th>THREAD</th>
<th>COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP BP R/W RATION (%)</td>
<td>GBP-DEPENDENT GETPAGES</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SYN.READS(XI)-DATA RETURNED</td>
<td>A</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SYN.READS(XI)-NO DATA RETURN</td>
<td>B</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SYN.READS(NF)-DATA RETURNED</td>
<td>C</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SYN.READS(NF)-NO DATA RETURN</td>
<td>D</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>UNREGISTER PAGE</td>
<td>E</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>CLEAN PAGES SYM.WRITTEN</td>
<td>F</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>REG.PAGE LIST (RPL) REQUEST</td>
<td>G</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>NUMBER OF PAGES RETR.FROM GBP</td>
<td>H</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PGS READ FRM DADS AFTER RPL</td>
<td>I</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ASYNC.READ-DATA RETURNED</td>
<td>J</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>PAGES CASTOUT</td>
<td>K</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>UNLOCK CASTOUT</td>
<td>L</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>READ CASTOUT CLASS</td>
<td>M</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>READ DIRECTORY INFO</td>
<td>N</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>READ STORAGE STATISTICS</td>
<td>O</td>
<td>5863.00</td>
<td>0.03</td>
</tr>
<tr>
<td>REG.PAGE</td>
<td>P</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>DELETE NAME</td>
<td>Q</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>ASYNCH GBP REQUESTS</td>
<td>R</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>EXPPLICIT X-INVALIDATIONS</td>
<td>S</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>CASTOUT CLASS THRESHOLD</td>
<td>T</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>GROUP BP CASTOUT THRESHOLD</td>
<td>U</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>GBP CHECKPOINTS TRIGGERED</td>
<td>V</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>CASTOUT ENGINE NOT AVAILABLE</td>
<td>W</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>WRITE ENGINE NOT AVAILABLE</td>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>READ FAILED-NO STORAGE</td>
<td>Y</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>WRITE FAILED-NO STORAGE</td>
<td>Z</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>WRITE TO SEC-GBP FAILED</td>
<td>A</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>DELETE NAME LIST SEC-GBP</td>
<td>B</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>DELETE NAME FROM SEC-GBP</td>
<td>C</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>UNLOCK CASTOUT STATS SEC-GBP</td>
<td>D</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>ASYNCH SEC-GBP REQUESTS</td>
<td>E</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>WRITE AND REGISTER</td>
<td>F</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>WRITE AND REGISTER MULT</td>
<td>G</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>CHANGED PGS SYNCRM.WRITN</td>
<td>H</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>CHANGED PGS ASYNCRM.WRITN</td>
<td>I</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PAGES WRITE &amp; REG MULT</td>
<td>J</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>READ FOR CASTOUT</td>
<td>K</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>READ FOR CASTOUT MULT</td>
<td>L</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>WRITE TO SEC-GBP</td>
<td>M</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>CLEAN PAGES ASYNCRM.WRITN</td>
<td>N</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>CLEAN PGS READ AFT.RPL</td>
<td>O</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PARTICIP.GBP REBUILD</td>
<td>P</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 70. Portion of OMEGAMON statistics detail report showing GBP activity

Explanation of fields:

A  The number of requests made to read a page from the group buffer pool
because the page was invalidated in the member’s buffer pool. The member found the required page in the group buffer pool.

**B**

The number of requests to read a page from the group buffer pool that were required because the page was invalidated in the member’s buffer pool. The member did not find the data in the group buffer pool and had to retrieve the page from DASD.

**C**

The number of requests made to read a page from the group buffer pool because the page was not in the member’s buffer pool. The member found the page in the group buffer pool.

**D**

The number of times DB2 unregistered interest for a single page. This happens when DB2 steals pages from the member’s buffer pool that belong to GBP-dependent page sets or partitions.

**E**

The number of data pages that were cast out of the member’s group buffer pool. Castout to a page set or partition is done by the castout owner of the page set or partition. This is normally the DB2 subsystem that had the first update intent on the page set or partition.

**F**

The number of times DB2 issued an unlock request to the coupling facility for completed castout I/Os. When pages are cast out to DASD, they are locked for castout in the coupling facility. This castout lock is not an IRLM lock; its function is to ensure that only one system can cast out a given page at a time.

**G**

The number of requests issued by the group buffer pool to determine which pages, from a particular page set or partition, must be cast out because they are cached as changed pages. This request is issued either by the page set or partition castout owner, or, when the group buffer pool castout threshold is reached by the group buffer pool structure owner.

**H**

The number of requests issued by the group buffer pool structure owner to read the directory entries of all changed pages in the group buffer pool. This request is issued at group buffer pool checkpoints to record the oldest recovery log record sequence number (LRSN). It is used as a basis for recovery if the group buffer pool fails. Such requests might have to be issued several times for each group buffer pool checkpoint to read the directory entries for all changed pages.

**I**

The number of times DB2 requested statistics information from the group buffer pool. It is issued by the group buffer pool structure owner at timed intervals to determine whether the group buffer pool castout threshold (GBPOOLT) has been reached.

**J**

The number of times DB2 registered interest in a single page. These are “register-only” requests, which means that DB2 is not requesting any data back from the request. This request is made only to create a directory entry for the page to be used for cross-invalidation when the page set or partition P-Lock is downgraded from S to IS mode, or from SIX to IX mode.

**K**

The number of requests made by DB2 to delete directory and data entries associated with a particular page set or partition from the group buffer pool. DB2 issues this request when it changes a page set or partition from GBP-dependent to non GBP-dependent. DB2 also issues this request for objects that are defined with GBPCACHE ALL when those objects are first opened.

**L**

The number of IXLCACHE invocations for the primary group buffer pool.
The number of times an explicit coupling facility cross-invalidation request was issued.

The number of times group buffer pool castout was initiated because the group buffer pool class castout threshold was detected.

The number of times a group buffer pool castout was initiated because the group buffer pool castout threshold was detected.

The number of group buffer pool checkpoints triggered by this member.

The number of times a castout engine was not available.

The number of times a coupling facility write engine was not available for coupling facility writes. This field is not applicable to versions higher than DB2 V7.

The number of coupling facility read requests that did not complete due to a lack of coupling facility storage resources. This field is not applicable for versions higher than DB2 V7.

The number of coupling facility write requests that could not complete due to a lack of coupling facility storage resources.

The number of coupling facility requests to read the castout statistics for the secondary group buffer pool. These requests are issued by the group buffer pool structure owner to check for orphaned data entries in the secondary group buffer pool.

The number of changed pages written synchronously to the group buffer pool. Pages are written with Write and Register (WAR) requests or Write and Register Multiple (WARM) requests. At commit time changed pages are forced from the member’s virtual buffer pool to the coupling facility.

The number of changed pages written asynchronously to the group buffer pool. Pages are written in response to Write and Register (WAR) and Write and Register Multiple (WARM) requests. Changed pages can be written from the member’s virtual buffer pool to the group coupling facility before the application commits. This happens when, for example, a local buffer pool threshold is reached, or when P-Lock negotiation forces the pages on the vertical deferred write queue to be written to the group buffer pool.

The number of coupling facility requests to write changed pages to the secondary group buffer pool for duplexing. This field is not applicable for versions higher than DB2 V7.

The number of clean pages that were asynchronously written to the group buffer pool from the virtual pool. This field is not applicable for versions higher than DB2 V7.

The number of times this member participated in a group buffer pool rebuild. This includes normal rebuilds and rebuilds to establish duplexing. This field is applicable for DB2 Versions 6 and 7 only.

Changing group buffer pools

The information under this heading, up to “Access path selection in a data sharing group” on page 272 is General-use Programming Interface and Associated Guidance Information, as defined in “Notices” on page 279.

This section describes how you can change attributes of the group buffer pool. The following tasks are described:

- “Changing the castout threshold values” on page 270
If you want to start or stop duplexing for a group buffer pool, see "Starting duplexing for a structure" on page 196 or "Stopping duplexing for a structure" on page 196. If you want to make hardware changes to the coupling facility or move a group buffer pool from one coupling facility to another, see "Shutting down the coupling facility" on page 197.

**Changing the castout threshold values**

Use the ALTER GROUPBUFFERPOOL command to change the group buffer pool castout thresholds. The following command, for example, changes the class castout threshold to 15% and the group buffer pool threshold to 55%:

```
-DB1A ALTER GROUPBUFFERPOOL(GBP1) CLASSST(15) GBPOOLT(55)
```

These changes take effect immediately.

**Changing the checkpoint frequency**

Use the ALTER GROUPBUFFERPOOL command to change the checkpoint frequency. For example, to indicate that you want group buffer pool checkpoints to occur every three minutes, enter the following command:

```
-DB1A ALTER GROUPBUFFERPOOL(GBP1) GBPCHKPT(3)
```

This change takes effect immediately.

**Changing the size of the group buffer pool**

You can use two methods to change the size of the group buffer pool. The method you choose depends on what level of coupling facility the group buffer pool is allocated and whether the group buffer pool is already allocated at the maximum size. For a duplexed group buffer pool, you are changing the size of both the primary and secondary structure with a single command.

**Dynamic method**: If all of the following conditions are true:

- The group buffer pool is allocated in a coupling facility with CFLEVEL greater than zero.
- The currently allocated size of the structure is less than the maximum size as defined in the SIZE parameter of the CFRM policy.

Then you can enter the following command (this example assumes the group name is DSND00A):  

```
SETXCF START,ALTER,STRNAME=DSND00A,GBPn,SIZE=newsize
```

This example assumes that *newsize* is less than or equal to the maximum size defined the CFRM policy for the group buffer pool.

If the maximum size (SIZE in the CFRM policy) is still not big enough, you must use the method described in "Static method" on page 271.

Assume a DISPLAY GROUPBUFFERPOOL command produced the following output:

```
DSNB757I -DB1A MVS CFRM POLICY STATUS FOR DSND00A_GBP1 = NORMAL
        MAX SIZE INDICATED IN POLICY = 4096 KB
```
And then you enter the following z/OS command to increase the size:

```
SETXCF START,ALTER,STRNM=DSNDB0A_GBP1,SIZE=1536
```

The DISPLAY GROUPBUFFERPOOL command output might look similar to the following after you alter the size:

```
DSN757I -A01A MVS CFRM POLICY STATUS FOR DSNDB0A_GBP1 = NORMAL
          MAX SIZE INDICATED IN POLICY = 4096 KB
          ALLOCATED = YES
DSN758I -A01A ALLOCATED SIZE = 1536 KB
          VOLATILITY STATUS = VOLATILE
          REBUILD STATUS = NONE
          DUPLEXING STATUS = SIMPLEXED
          CFNAME = IFL01
          OPERATIONAL CFLEVEL = 5
          ACTUAL CFLEVEL = 7
DSN759I -A01A NUMBER OF DIRECTORY ENTRIES = 924
          NUMBER OF DATA PAGES = 180
          NUMBER OF CONNECTIONS = 2
```

Notice that the allocated size, the numbers of directory entries, and the number of data pages has increased. The existing ratio is maintained.

**Static method:** If any of the following conditions are true:

- The group buffer pool is allocated in a coupling facility at CFLEVEL=0.
- The allocated size of the structure is already at the maximum size defined by the SIZE parameter of the CFRM policy.

You must use the following procedure. Because the group buffer pool must be rebuilt, use this procedure when there is little activity in the group.

1. Increase the storage for the group buffer pool in the CFRM policy, and use DUPLEX(ENABLED).
2. Use the following z/OS command to start the updated policy:
   ```
   SETXCF START,POLICY,TYPE=CFRM,POLNAME=policyname
   ```
3. Use the following command to stop duplexing:
   ```
   SETXCF STOP,REBUILD,DUPLEX,STRNAME=strname,KEEP=OLD
   ```
4. Use the following command to rebuild the group buffer pool:
   ```
   SETXCF START,REBUILD,STRNAME=strname
   ```
5. Use the DISPLAY GROUPBUFFERPOOL command to determine whether the rebuild caused duplexing to restart.
6. If duplexing did not restart on its own, use the following command to restart it:
   ```
   SETXCF START,REBUILD,DUPLEX,STRNAME=strname
   ```
Changing the ratio of directory to data entries

To change the ratio of directory to data entries, you must use the ALTER GROUPBUFFERPOOL command. For example, if the current ratio is 5 (that is, there are five directory entries to every one data page), you can use an ALTER GROUPBUFFERPOOL command, similar to the one shown here, to increase the ratio to 7:

```
-DB1A ALTER GROUPBUFFERPOOL (GBP0) RATIO (7)
```

For the change to take effect, you must rebuild the group buffer pool by using the SETXCF START, REBUILD command.

For duplexed group buffer pools, you must stop duplexing before you can rebuild the group buffer pool. Use the following procedure to rebuild duplexed group buffer pools:

1. Stop duplexing, as described in “Stopping duplexing for a structure” on page 196 to revert the group buffer pool to simplex mode.
2. Alter the group buffer pool ratio and issue the SETXCF START,REBUILD,STRNAME=strname command.
3. Start duplexing, as described in “Starting duplexing for a structure” on page 196.

Changing the GBPCACHE attribute

To change the GBPCACHE option, use the ALTER GROUPBUFFERPOOL command. For the change to take effect, you must rebuild the group buffer pool by using the SETXCF START,REBUILD,STRNAME=strname command.

DB2 does let you duplex a GBPCACHE(NO) group buffer pool. If the group buffer pool is currently GBPCACHE(YES) but the NO attribute is pending, DB2 ignores the pending GBPCACHE(NO) attribute if you start a duplexing rebuild.

Access path selection in a data sharing group

This section describes the following topics:

* “Effect of member configuration on access path selection”
* “How EXPLAIN works in a data sharing group” on page 273

Effect of member configuration on access path selection

Because plans and packages are bound on individual members in the group, the way a member is configured influences the access path chosen for statements in that plan or package. For example, it is possible to have different buffer pool sizes and different RID (record identifier) pool sizes on each member. It is also possible that members are on different CPC models.

When you bind your application from one of the members, DB2 chooses the best access path, given the catalog statistics, CPC model and buffer pool sizes, among other things. Suppose, though, that the selected access path is optimal for the one member, but is a relatively poor choice for a different member in the same group. Because the group shares the catalog and directory, the same plan (and hence the same access paths) are used regardless of member, after the application is bound.

Where to bind in a mixed data sharing configuration: If your data sharing group consists of mixed CPC models, be aware that the speed of a central processor might change your access path. This effect is more likely to occur with long-running queries than with fast-running transactions.
Automatic rebind: The access path can change if automatic rebind occurs while the application is executing on a different member than the one on which the original bind occurred.

How EXPLAIN works in a data sharing group

EXPLAIN informs you about access paths that DB2 chooses. Because EXPLAIN can be run on one member and a plan can be bound and executed on other members in a data sharing group, it is important to know which member performed the EXPLAIN. The PLAN_TABLE’s GROUP_MEMBER column contains the name of the member that performed the EXPLAIN. The column is blank if the member was not in a data sharing environment when the EXPLAIN was performed.

Maintaining in-memory statistics in data sharing

To enable automation of utility job scheduling, DB2 collects statistics for use by users or automated task schedulers to determine which objects might require REORG, RUNSTATS, or COPY. These statistics are collected in real time and periodically written to DB2 tables.

Statistics are maintained for each partition if the page set is partitioned. To externalize the statistics from in-memory to the real-time statistics tables, DB2 examines the in-memory statistics, calculates the new totals, updates the real-time statistics tables, and then resets the in-memory statistics. This process is an asynchronous task, managed by the statistics manager.

In DB2 data sharing, each member updates their statistics in a serial process. Each member reads the target row from the statistics table, and while holding a lock, aggregates their in-memory statistics and then updates the statistics tables with the new totals.

Utilities that reset page sets to empty can invalidate the in-memory statistics of other members. COPY and RUNSTATS must externalize other members’ statistics before running. The member that is running the utility notifies the other members, and the in-memory statistics are either invalidated or externalized. If the notify process fails, the running utility will not fail. The appropriate timestamp (ReorgLastTime, StatsLastTime, or CopyLastTime) is set to NULL to indicate that the table statistics are unknown.

See Appendix G (Volume 2) of DB2 Administration Guide or Appendix F of DB2 Utility Guide and Reference for more information.
Appendix. DB2 and IRLM names

This appendix includes information about name formats for the licensed program DB2 UDB for z/OS and its IRLM.

The following topics provide additional information:
- “DB2 group names”
- “DB2 member names”
- “IRLM names” on page 276

DB2 group names

DB2 group names include the generic LU name, group attachment name, location name, and coupling facility structure names.

Table 57. Group names. One set of names exists per group.

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>Example format</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog alias</td>
<td>8</td>
<td>catalias</td>
<td>You must place the catalog alias for DB2 in the z/OS master catalog.</td>
</tr>
<tr>
<td>Catalog and directory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>database names</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSNHDECIP</td>
<td>8</td>
<td>DSNHDECIP</td>
<td>All DB2 subsystems in the group share the same catalog and directory.</td>
</tr>
<tr>
<td>Generic LU name</td>
<td>8</td>
<td>xxxxxxxx</td>
<td>One name per group. Requesters use this name to configure their communications control information.</td>
</tr>
<tr>
<td>Group attachment name</td>
<td>4</td>
<td>gssn</td>
<td>This name can be used by TSO/batch, CAF, RRSAF, and utilities as a generic attachment name.</td>
</tr>
<tr>
<td>Group buffer pools</td>
<td>16</td>
<td>groupname_GBPxxxx</td>
<td>You must enter this name on the CFM policy.</td>
</tr>
<tr>
<td>Group buffer pools (coupling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>facility cache structures)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group name</td>
<td>8</td>
<td>xxxxxxxx</td>
<td>This name must be unique within the Parallel Sysplex.</td>
</tr>
<tr>
<td>Location name</td>
<td>16</td>
<td>xxxxxxxxxxxxxxxxxx</td>
<td>One name per group. Requesters use this name in their SQL applications.</td>
</tr>
<tr>
<td>Lock structure</td>
<td>16</td>
<td>groupname_LOCK1</td>
<td>You must enter this name on the CFM policy.</td>
</tr>
<tr>
<td>SCA (coupling facility</td>
<td>16</td>
<td>groupname_SCA</td>
<td>You must enter this name on the CFM policy.</td>
</tr>
<tr>
<td>list structure)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target libraries</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>catalias.SDSNCLST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>catalias.SDSNLINK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>catalias.SDSNLOAD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>catalias.SDSNEXIT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target libraries can be shared</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>among DB2 group members or can</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>be replicated.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DB2 member names

Member names include the individual DB2 member (subsystem) name and its associated z/OS subsystem name, procedure names, and BSDS and log names.
Table 58. Member names. One set of names exists per member.

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>Example format</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active log prefixes</td>
<td>30</td>
<td>membname.LOGCOPY1</td>
<td>The catalog alias can be added as first qualifier.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>membname.LOGCOPY2</td>
<td></td>
</tr>
<tr>
<td>Archive log prefixes</td>
<td>19 or 35</td>
<td>membname.ARCLG1</td>
<td>The catalog alias can be added as the first qualifier. Only 19 characters can be used for the prefix if the name is timestamped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>membname.ARCLG2</td>
<td></td>
</tr>
<tr>
<td>BSDS</td>
<td>33</td>
<td>membname.BSDS01</td>
<td>The catalog alias can be added as the first qualifier.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>membname.BSDS02</td>
<td></td>
</tr>
<tr>
<td>SCA and group buffer pool connection names</td>
<td>16</td>
<td>DB2_membername</td>
<td>The connection name is generated by DB2. You see it only in certain commands (z/OS D XCF, STRUCTURE and the DB2 connection list display of DISPLAY GROUPBUFFERPOOL).</td>
</tr>
<tr>
<td>Command prefix</td>
<td>8</td>
<td>xmembcgf</td>
<td>The first character must be a special character.</td>
</tr>
<tr>
<td>LU name</td>
<td>8</td>
<td>luname</td>
<td>This name must be unique within the data sharing group and the network.</td>
</tr>
<tr>
<td>Member name</td>
<td>8</td>
<td>membname</td>
<td>This name must be unique within the data sharing group. If the member name is also used as the high level qualifier for the member's data sets (BSDS, logs, and so on), member names must be unique within the z/OS Parallel Sysplex. This is because the z/OS Parallel Sysplex can have a shared master catalog.</td>
</tr>
<tr>
<td>Procedure names</td>
<td>8</td>
<td>mssnMSTR</td>
<td>These names are generated from the member subsystem name at installation time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mssnDBM1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>mssnDIST</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>mssnSPAS</td>
<td></td>
</tr>
<tr>
<td>Subsystem name</td>
<td>4</td>
<td>mssn</td>
<td>This name is used by all the attachment interfaces and must be unique within the z/OS Parallel Sysplex.</td>
</tr>
<tr>
<td>Subsystem parameters load module</td>
<td>8</td>
<td>DSNZPxxx</td>
<td>Resides in SDSNEXIT. The name is specified as a parameter on mssnMSTR procedure.</td>
</tr>
<tr>
<td>XCF member name</td>
<td>16</td>
<td>membname</td>
<td>This is the same as the member name for DB2; thus, a maximum of eight characters is used.</td>
</tr>
<tr>
<td>Work file database</td>
<td>8</td>
<td>mworkdb</td>
<td>Work file data sets have names of the format catalias.DSNDBC.mworkdb.DSNknn.y0001.A001, where y can be either I or J.</td>
</tr>
</tbody>
</table>

IRLM names

Each DB2 subsystem in the data sharing group has an associated IRLM. Each member’s IRLM and its associated procedures must be named. The IRLM group name, subsystem name, and member ID are parameters on the IRLM procedure. This requires a separate IRLM procedure for every IRLM in the group.

Table 59. IRLM names. One set of names exists per DB2 member.

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>Example format</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRLM group name</td>
<td>8</td>
<td>xxxxxxx</td>
<td>This is a parameter on iss0IRLM procedure. It must be unique across the Parallel Sysplex, to avoid name conflicts.</td>
</tr>
</tbody>
</table>
Table 59. IRLM names (continued). One set of names exists per DB2 member.

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>Example format</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRLM subsystem name</td>
<td>4</td>
<td>issn</td>
<td>This is a parameter on the issn IRLM procedure. This subsystem name must be unique within the data sharing group to avoid the problem of DB2 connecting to the wrong IRLM.</td>
</tr>
<tr>
<td>IRLM procedure name</td>
<td>8</td>
<td>mssnIRLM</td>
<td>Each DB2 member knows its IRLM by the procedure and subsystem name saved in that member's subsystem parameter load module.</td>
</tr>
<tr>
<td>IRLM member ID</td>
<td>3</td>
<td></td>
<td>This ID uniquely names an IRLM within a group. It is a parameter on issnIRLM procedure and must be unique within the data sharing group.</td>
</tr>
<tr>
<td>IRLM XCF member name</td>
<td>16</td>
<td>xxxxxxxx$issnNNN</td>
<td>xxxxxxxx is the IRLM group name, issn is the IRLM subsystem ID, and NNN is the IRLM member ID. Dollar signs ($) are used as padding. This name is generated at startup time.</td>
</tr>
<tr>
<td>Lock structure connection name</td>
<td>16</td>
<td>xxxxxxxx$issnNNN</td>
<td>The connection name is the same as the IRLM XCF member name. It is generated by IRLM. You see it only as the output of certain commands (such as z/OS D XCF,STRUCTURE). In some cases, the connection name can be of the format xxxxxxxx#issnNNN.</td>
</tr>
</tbody>
</table>
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Programming interface information

This book is intended to help you plan for the use of DB2 data sharing.

This book also documents General-use Programming Interface and Associated Guidance Information and Product-sensitive Programming Interface and Associated Guidance Information provided by DB2 UDB for z/OS (DB2).

General-use programming interfaces allow the customer to write programs that obtain the services of DB2.

General-use Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

<table>
<thead>
<tr>
<th>General-use programming interface</th>
</tr>
</thead>
</table>

General-use Programming Interface and Associated Guidance Information ...

<table>
<thead>
<tr>
<th>End of General-use programming interface</th>
</tr>
</thead>
</table>
Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of DB2. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces might need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs, by the following marking:

Product-sensitive programming interface

End of Product-sensitive programming interface

**Trademarks**

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

- BookManager
- CICS
- CICSPlex
- CT
- DataPropagator
- DB2
- DB2 Connect
- DB2 Universal Database
- DFSMSdfp
- DFSMSdss
- DFSMSShsm
- Distributed Relational Database Architecture
- DRDA
- Enterprise Storage Server
- ES/3090
- eServer
- FlashCopy
- IBM
- IBM Registry
- ibm.com
- IMS
- iSeries
- Language Environment
- MVS
- OS/390
- Parallel Sysplex
- PR/SM
- QMF
- RACF
- RAMAC
- Redbooks
- RMF
- Sysplex Timer
- System/390
- TotalStorage
- UNIX
- VTAM
- WebSphere
- z/OS
- zSeries

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.
Glossary

The following terms and abbreviations are defined as they are used in the DB2 library.

A

abend. Abnormal end of task. 

abend reason code. A 4-byte hexadecimal code that uniquely identifies a problem with DB2.

abnormal end of task (abend). Termination of a task, job, or subsystem because of an error condition that recovery facilities cannot resolve during execution.

access method services. The facility that is used to define and reproduce VSAM key-sequenced data sets.

access path. The path that is used to locate data that is specified in SQL statements. An access path can be indexed or sequential.

active log. The portion of the DB2 log to which log records are written as they are generated. The active log always contains the most recent log records, whereas the archive log holds those records that are older and no longer fit on the active log.

active member state. A state of a member of a data sharing group. The cross-system coupling facility identifies each active member with a group and associates the member with a particular task, address space, and z/OS system. A member that is not active has either a failed member state or a quiesced member state.

address space. A range of virtual storage pages that is identified by a number (ASID) and a collection of segment and page tables that map the virtual pages to real pages of the computer's memory.

address space connection. The result of connecting an allied address space to DB2. Each address space that contains a task that is connected to DB2 has exactly one address space connection, even though more than one task control block (TCB) can be present. See also allied address space and task control block.

address space identifier (ASID). A unique system-assigned identifier for an address space.

administrative authority. A set of related privileges that DB2 defines. When you grant one of the administrative authorities to a person's ID, the person has all of the privileges that are associated with that administrative authority.

after trigger. A trigger that is defined with the trigger activation time AFTER.

agent. As used in DB2, the structure that associates all processes that are involved in a DB2 unit of work. An allied agent is generally synonymous with an allied thread. System agents are units of work that process tasks that are independent of the allied agent, such as prefetch processing, deferred writes, and service tasks.

# aggregate function. An operation that derives its result by using values from one or more rows. Contrast with scalar function.

alias. An alternative name that can be used in SQL statements to refer to a table or view in the same or a remote DB2 subsystem.

allied address space. An area of storage that is external to DB2 and that is connected to DB2. An allied address space is capable of requesting DB2 services.

allied thread. A thread that originates at the local DB2 subsystem and that can access data at a remote DB2 subsystem.

allocated cursor. A cursor that is defined for stored procedure result sets by using the SQL ALLOCATE CURSOR statement.

already verified. An LU 6.2 security option that allows DB2 to provide the user's verified authorization ID when allocating a conversation. With this option, the user is not validated by the partner DB2 subsystem.

ambiguous cursor. A database cursor that is in a plan or package that contains either PREPARE or EXECUTE IMMEDIATE SQL statements, and for which the following statements are true: the cursor is not defined with the FOR READ ONLY clause or the FOR UPDATE OF clause; the cursor is not defined on a read-only result table; the cursor is not the target of a WHERE CURRENT clause on an SQL UPDATE or DELETE statement.

American National Standards Institute (ANSI). An organization consisting of producers, consumers, and general interest groups, that establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States.

ANSI. American National Standards Institute.

APAR. Authorized program analysis report.

APAR fix corrective service. A temporary correction of an IBM software defect. The correction is temporary,
because it is usually replaced at a later date by a more permanent correction, such as a program temporary fix (PTF).

**APF.** Authorized program facility.

**API.** Application programming interface.

**APPL.** A VTAM network definition statement that is used to define DB2 to VTAM as an application program that uses SNA LU 6.2 protocols.

**application.** A program or set of programs that performs a task; for example, a payroll application.

**application-directed connection.** A connection that an application manages using the SQL CONNECT statement.

**application plan.** The control structure that is produced during the bind process. DB2 uses the application plan to process SQL statements that it encounters during statement execution.

**application process.** The unit to which resources and locks are allocated. An application process involves the execution of one or more programs.

**application programming interface (API).** A functional interface that is supplied by the operating system or by a separately orderable licensed program that allows an application program that is written in a high-level language to use specific data or functions of the operating system or licensed program.

**application requester.** The component on a remote system that generates DRDA requests for data on behalf of an application. An application requester accesses a DB2 database server using the DRDA application-directed protocol.

**application server.** The target of a request from a remote application. In the DB2 environment, the application server function is provided by the distributed data facility and is used to access DB2 data from remote applications.

**archive log.** The portion of the DB2 log that contains log records that have been copied from the active log.

**ASCII.** An encoding scheme that is used to represent strings in many environments, typically on PCs and workstations. Contrast with EBCDIC and Unicode.

**ASID.** Address space identifier.

**attachment facility.** An interface between DB2 and TSO, IMS, CICS, or batch address spaces. An attachment facility allows application programs to access DB2.

**attribute.** A characteristic of an entity. For example, in database design, the phone number of an employee is one of that employee’s attributes.

**authorization ID.** A string that can be verified for connection to DB2 and to which a set of privileges is allowed. It can represent an individual, an organizational group, or a function, but DB2 does not determine this representation.

**authorized program analysis report (APAR).** A report of a problem that is caused by a suspected defect in a current release of an IBM supplied program.

**authorized program facility (APF).** A facility that permits the identification of programs that are authorized to use restricted functions.

**automatic query rewrite.** A process that examines an SQL statement that refers to one or more base tables, and, if appropriate, rewrites the query so that it performs better. This process can also determine whether to rewrite a query so that it refers to one or more materialized query tables that are derived from the source tables.

**auxiliary index.** An index on an auxiliary table in which each index entry refers to a LOB.

**auxiliary table.** A table that stores columns outside the table in which they are defined. Contrast with base table.

**B**

**backout.** The process of undoing uncommitted changes that an application process made. This might be necessary in the event of a failure on the part of an application process, or as a result of a deadlock situation.

**backward log recovery.** The fourth and final phase of restart processing during which DB2 scans the log in a backward direction to apply UNDO log records for all aborted changes.

**base table.** (1) A table that is created by the SQL CREATE TABLE statement and that holds persistent data. Contrast with result table and temporary table.

(2) A table containing a LOB column definition. The actual LOB column data is not stored with the base table. The base table contains a row identifier for each row and an indicator column for each of its LOB columns. Contrast with auxiliary table.

**base table space.** A table space that contains base tables.

**basic predicate.** A predicate that compares two values.

**basic sequential access method (BSAM).** An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential-access or a direct-access device.
batch message processing program.  In IMS, an application program that can perform batch-type processing online and can access the IMS input and output message queues.

before trigger.  A trigger that is defined with the trigger activation time BEFORE.

binary integer.  A basic data type that can be further classified as small integer or large integer.

# binary large object (BLOB).  A sequence of bytes in which the size of the value ranges from 0 bytes to 2 GB−1. Such a string has a CCSID value of 65535.

binary string.  A sequence of bytes that is not associated with a CCSID. For example, the BLOB data type is a binary string.

bind.  The process by which the output from the SQL precompiler is converted to a usable control structure, often called an access plan, application plan, or package. During this process, access paths to the data are selected and some authorization checking is performed. The types of bind are:

automatic bind.  (More correctly, automatic rebind) A process by which SQL statements are bound automatically (without a user issuing a BIND command) when an application process begins execution and the bound application plan or package it requires is not valid.

dynamic bind.  A process by which SQL statements are bound as they are entered.

incremental bind.  A process by which SQL statements are bound during the execution of an application process.

static bind.  A process by which SQL statements are bound after they have been precompiled. All static SQL statements are prepared for execution at the same time.

# bit data.  Data that is character type CHAR or VARCHAR and has a CCSID value of 65535.

BLOB.  Binary large object.

block fetch.  A capability in which DB2 can retrieve, or fetch, a large set of rows together. Using block fetch can significantly reduce the number of messages that are being sent across the network. Block fetch applies only to cursors that do not update data.

BMP.  Batch Message Processing (IMS). See batch message processing program.

bootstrap data set (BSDS).  A VSAM data set that contains name and status information for DB2, as well as RBA range specifications, for all active and archive log data sets. It also contains passwords for the DB2 directory and catalog, and lists of conditional restart and checkpoint records.

BSAM.  Basic sequential access method.

BSDS.  Bootstrap data set.

buffer pool.  Main storage that is reserved to satisfy the buffering requirements for one or more table spaces or indexes.

built-in data type.  A data type that IBM supplies. Among the built-in data types for DB2 UDB for z/OS are string, numeric, ROWID, and datetime. Contrast with distinct type.

built-in function.  A function that DB2 supplies. Contrast with user-defined function.

business dimension.  A category of data, such as products or time periods, that an organization might want to analyze.

C

cache structure.  A coupling facility structure that stores data that can be available to all members of a Sysplex. A DB2 data sharing group uses cache structures as group buffer pools.

CAF.  Call attachment facility.

call attachment facility (CAF).  A DB2 attachment facility for application programs that run in TSO or z/OS batch. The CAF is an alternative to the DSN command processor and provides greater control over the execution environment.

call-level interface (CLI).  A callable application programming interface (API) for database access, which is an alternative to using embedded SQL. In contrast to embedded SQL, DB2 ODBC (which is based on the CLI architecture) does not require the user to precompile or bind applications, but instead provides a standard set of functions to process SQL statements and related services at run time.

cascade delete.  The way in which DB2 enforces referential constraints when it deletes all descendent rows of a deleted parent row.

CASE expression.  An expression that is selected based on the evaluation of one or more conditions.

cast function.  A function that is used to convert instances of a (source) data type into instances of a different (target) data type. In general, a cast function has the name of the target data type. It has one single argument whose type is the source data type; its return type is the target data type.

castout.  The DB2 process of writing changed pages from a group buffer pool to disk.

castout owner.  The DB2 member that is responsible for casting out a particular page set or partition.
catalog • closed application

catalog. In DB2, a collection of tables that contains descriptions of objects such as tables, views, and indexes.

catalog table. Any table in the DB2 catalog.

CCSID. Coded character set identifier.

CDB. Communications database.

CDRA. Character Data Representation Architecture.

CEC. Central electronic complex. See central processor complex.


central processor (CP). The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load, and other machine operations.

central processor complex (CPC). A physical collection of hardware (such as an ES/3090™) that consists of main storage, one or more central processors, timers, and channels.

CFRM. Coupling facility resource management.

CFRM policy. A declaration by a z/OS administrator regarding the allocation rules for a coupling facility structure.

character conversion. The process of changing characters from one encoding scheme to another.

Character Data Representation Architecture (CDRA). An architecture that is used to achieve consistent representation, processing, and interchange of string data.

character large object (CLOB). A sequence of bytes representing single-byte characters or a mixture of single- and double-byte characters where the size of the value can be up to 2 GB−1. In general, character large object values are used whenever a character string might exceed the limits of the VARCHAR type.

character set. A defined set of characters.

character string. A sequence of bytes that represent bit data, single-byte characters, or a mixture of single-byte and multibyte characters.

check constraint. A user-defined constraint that specifies the values that specific columns of a base table can contain.

check integrity. The condition that exists when each row in a table conforms to the check constraints that are defined on that table. Maintaining check integrity requires DB2 to enforce check constraints on operations that add or change data.

check pending. A state of a table space or partition that prevents its use by some utilities and by some SQL statements because of rows that violate referential constraints, check constraints, or both.

checkpoint. A point at which DB2 records internal status information on the DB2 log; the recovery process uses this information if DB2 abnormally terminates.

child lock. For explicit hierarchical locking, a lock that is held on either a table, page, row, or a large object (LOB). Each child lock has a parent lock. See also parent lock.

CI. Control interval.

CICS. Represents (in this publication): CICS Transaction Server for z/OS: Customer Information Control System Transaction Server for z/OS.

CICS attachment facility. A DB2 subcomponent that uses the z/OS subsystem interface (SSI) and cross-storage linkage to process requests from CICS to DB2 and to coordinate resource commitment.

CIDF. Control interval definition field.

claim. A notification to DB2 that an object is being accessed. Claims prevent drains from occurring until the claim is released, which usually occurs at a commit point. Contrast with drain.

claim class. A specific type of object access that can be one of the following isolation levels:

Cursor stability (CS)
Repeatable read (RR)
Write

claim count. A count of the number of agents that are accessing an object.

class of service. A VTAM term for a list of routes through a network, arranged in an order of preference for their use.

class word. A single word that indicates the nature of a data attribute. For example, the class word PROJ indicates that the attribute identifies a project.

clause. In SQL, a distinct part of a statement, such as a SELECT clause or a WHERE clause.

CLI. Call-level interface.

client. See requester.

CLIST. Command list. A language for performing TSO tasks.

CLOB. Character large object.

closed application. An application that requires exclusive use of certain statements on certain DB2
objects, so that the objects are managed solely through
the application’s external interface.

**CLPA.** Create link pack area.

- **clustering index.** An index that determines how rows
  are physically ordered (clustered) in a table space. If a
  clustering index on a partitioned table is not a
  partitioning index, the rows are ordered in cluster
  sequence within each data partition instead of spanning
  partitions. Prior to Version 8 of DB2 UDB for z/OS, the
  partitioning index was required to be the clustering
  index.

- **coded character set.** A set of unambiguous rules that
  establish a character set and the one-to-one
  relationships between the characters of the set and their
  coded representations.

- **coded character set identifier (CCSID).** A 16-bit
  number that uniquely identifies a coded representation
  of graphic characters. It designates an encoding scheme
  identifier and one or more pairs consisting of a
  character set identifier and an associated code page
  identifier.

- **code page.** (1) A set of assignments of characters to
  code points. In EBCDIC, for example, the character ‘A’
  is assigned code point ‘X’C1’ (2), and character ‘B’ is
  assigned code point ‘XC2’. Within a code page, each
  code point has only one specific meaning.

- **code point.** In CDRA, a unique bit pattern that
  represents a character in a code page.

# code unit. The fundamental binary width in a
# computer architecture that is used for representing
# character data, such as 7 bits, 8 bits, 16 bits, or 32 bits.
# Depending on the character encoding form that is used,
# each code point in a coded character set can be
# represented internally by one or more code units.

- **coexistence.** During migration, the period of time in
  which two releases exist in the same data sharing
  group.

- **cold start.** A process by which DB2 restarts without
  processing any log records. Contrast with warm start.

- **collection.** A group of packages that have the same
  qualifier.

- **column.** The vertical component of a table. A column
  has a name and a particular data type (for example,
  character, decimal, or integer).

# **column function.** See aggregate function.

- **"come from" checking.** An LU 6.2 security option that
  defines a list of authorization IDs that are allowed to
  connect to DB2 from a partner LU.

- **command.** A DB2 operator command or a DSN
  subcommand. A command is distinct from an SQL
  statement.

- **command prefix.** A one- to eight-character command
  identifier. The command prefix distinguishes the
  command as belonging to an application or subsystem
  rather than to MVS.

- **command recognition character (CRC).** A character
  that permits a z/OS console operator or an IMS
  subsystem user to route DB2 commands to specific DB2
  subsystems.

- **command scope.** The scope of command operation in
  a data sharing group. If a command has member scope,
  the command displays information only from the one
  member or affects only non-shared resources that are
  owned locally by that member. If a command has group
  scope, the command displays information from all
  members, affects non-shared resources that are owned
  locally by all members, displays information on
  sharable resources, or affects sharable resources.

- **commit.** The operation that ends a unit of work by
  releasing locks so that the database changes that are
  made by that unit of work can be perceived by other
  processes.

- **commit point.** A point in time when data is
  considered consistent.

- **committed phase.** The second phase of the multisite
  update process that requests all participants to commit
  the effects of the logical unit of work.

- **common service area (CSA).** In z/OS, a part of the
  common area that contains data areas that are
  addressable by all address spaces.

- **communications database (CDB).** A set of tables in
  the DB2 catalog that are used to establish conversations
  with remote database management systems.

- **comparison operator.** A token (such as =, >, or <) that
  is used to specify a relationship between two values.

- **composite key.** An ordered set of key columns of the
  same table.

- **compression dictionary.** The dictionary that controls
  the process of compression and decompression. This
  dictionary is created from the data in the table space or
  table space partition.

- **concurrency.** The shared use of resources by more
  than one application process at the same time.

- **conditional restart.** A DB2 restart that is directed by a
  user-defined conditional restart control record (CRCR).

- **connection.** In SNA, the existence of a communication
  path between two partner LUs that allows information
connection context • coupling facility

to be exchanged (for example, two DB2 subsystems that are connected and communicating by way of a conversation).

connection context. In SQLJ, a Java™ object that represents a connection to a data source.

connection declaration clause. In SQLJ, a statement that declares a connection to a data source.

connection handle. The data object containing information that is associated with a connection that DB2 ODBC manages. This includes general status information, transaction status, and diagnostic information.

connection ID. An identifier that is supplied by the attachment facility and that is associated with a specific address space connection.

consistency token. A timestamp that is used to generate the version identifier for an application. See also version.

constant. A language element that specifies an unchanging value. Constants are classified as string constants or numeric constants. Contrast with variable.

constraint. A rule that limits the values that can be inserted, deleted, or updated in a table. See referential constraint, check constraint, and unique constraint.

context. The application’s logical connection to the data source and associated internal DB2 ODBC connection information that allows the application to direct its operations to a data source. A DB2 ODBC context represents a DB2 thread.

contracting conversion. A process that occurs when the length of a converted string is smaller than that of the source string. For example, this process occurs when an EBCDIC mixed-data string that contains DBCS characters is converted to ASCII mixed data; the converted string is shorter because of the removal of the shift codes.

control interval (CI). A fixed-length area or disk in which VSAM stores records and creates distributed free space. Also, in a key-sequenced data set or file, the set of records that an entry in the sequence-set index record points to. The control interval is the unit of information that VSAM transmits to or from disk. A control interval always includes an integral number of physical records.

control interval definition field (CIDF). In VSAM, a field that is located in the 4 bytes at the end of each control interval; it describes the free space, if any, in the control interval.

conversation. Communication, which is based on LU 6.2 or Advanced Program-to-Program Communication (APPC), between an application and a remote transaction program over an SNA logical unit-to-logical unit (LU-LU) session that allows communication while processing a transaction.

coordinator. The system component that coordinates the commit or rollback of a unit of work that includes work that is done on one or more other systems.

copy pool. A named set of SMS storage groups that contains data that is to be copied collectively. A copy pool is an SMS construct that lets you define which storage groups are to be copied by using FlashCopy® functions. HSM determines which volumes belong to a copy pool.

copy target. A named set of SMS storage groups that are to be used as containers for copy pool volume copies. A copy target is an SMS construct that lets you define which storage groups are to be used as containers for volumes that are copied by using FlashCopy functions.

copy version. A point-in-time FlashCopy copy that is managed by HSM. Each copy pool has a version parameter that specifies how many copy versions are maintained on disk.

correlated columns. A relationship between the value of one column and the value of another column.

correlated subquery. A subquery (part of a WHERE or HAVING clause) that is applied to a row or group of rows of a table or view that is named in an outer subselect statement.

correlation ID. An identifier that is associated with a specific thread. In TSO, it is either an authorization ID or the job name.

correlation name. An identifier that designates a table, a view, or individual rows of a table or view within a single SQL statement. It can be defined in any FROM clause or in the first clause of an UPDATE or DELETE statement.

cost category. A category into which DB2 places cost estimates for SQL statements at the time the statement is bound. A cost estimate can be placed in either of the following cost categories:

- A: Indicates that DB2 had enough information to make a cost estimate without using default values.
- B: Indicates that some condition exists for which DB2 was forced to use default values for its estimate.

The cost category is externalized in the COSTCATEGORY column of the DSN_STATEMNT_TABLE when a statement is explained.

coupling facility. A special PR/SM™ LPAR logical partition that runs the coupling facility control program and provides high-speed caching, list processing, and locking functions in a Parallel Sysplex.
**coupling facility resource management.** A component of z/OS that provides the services to manage coupling facility resources in a Parallel Sysplex. This management includes the enforcement of CFRM policies to ensure that the coupling facility and structure requirements are satisfied.

**CP.** Central processor.

**CPC.** Central processor complex.

**C++ member.** A data object or function in a structure, union, or class.

**C++ member function.** An operator or function that is declared as a member of a class. A member function has access to the private and protected data members and to the member functions of objects in its class. Member functions are also called methods.

**C++ object.** (1) A region of storage. An object is created when a variable is defined or a new function is invoked. (2) An instance of a class.

**CRC.** Command recognition character.

**CRCR.** Conditional restart control record. See also conditional restart.

**create link pack area (CLPA).** An option that is used during IPL to initialize the link pack pageable area.

**created temporary table.** A table that holds temporary data and is defined with the SQL statement CREATE GLOBAL TEMPORARY TABLE. Information about created temporary tables is stored in the DB2 catalog, so this kind of table is persistent and can be shared across application processes. Contrast with declared temporary table. See also temporary table.

**cross-memory linkage.** A method for invoking a program in a different address space. The invocation is synchronous with respect to the caller.

**cross-system coupling facility (XCF).** A component of z/OS that provides functions to support cooperation between authorized programs that run within a Sysplex.

**cross-system extended services (XES).** A set of z/OS services that allow multiple instances of an application or subsystem, running on different systems in a Sysplex environment, to implement high-performance, high-availability data sharing by using a coupling facility.

**CS.** Cursor stability.

**CSA.** Common service area.

**CT.** Cursor table.

**current data.** Data within a host structure that is current with (identical to) the data within the base table.

**current SQL ID.** An ID that, at a single point in time, holds the privileges that are exercised when certain dynamic SQL statements run. The current SQL ID can be a primary authorization ID or a secondary authorization ID.

**current status rebuild.** The second phase of restart processing during which the status of the subsystem is reconstructed from information on the log.

**cursor.** A named control structure that an application program uses to point to a single row or multiple rows within some ordered set of rows of a result table. A cursor can be used to retrieve, update, or delete rows from a result table.

**cursor sensitivity.** The degree to which database updates are visible to the subsequent FETCH statements in a cursor. A cursor can be sensitive to changes that are made with positioned update and delete statements specifying the name of that cursor. A cursor can also be sensitive to changes that are made with searched update or delete statements, or with cursors other than this cursor. These changes can be made by this application process or by another application process.

**cursor stability (CS).** The isolation level that provides maximum concurrency without the ability to read uncommitted data. With cursor stability, a unit of work holds locks only on its uncommitted changes and on the current row of each of its cursors.

**cursor table (CT).** The copy of the skeleton cursor table that is used by an executing application process.

**cycle.** A set of tables that can be ordered so that each table is a descendent of the one before it, and the first table is a descendent of the last table. A self-referencing table is a cycle with a single member.

**D**

**DAD.** See Document access definition.

**disk.** A direct-access storage device that records data magnetically.

**database.** A collection of tables, or a collection of table spaces and index spaces.

**database access thread.** A thread that accesses data at the local subsystem on behalf of a remote subsystem.

**database administrator (DBA).** An individual who is responsible for designing, developing, operating, safeguarding, maintaining, and using a database.
**database alias • DBA**

- **database alias.** The name of the target server if different from the location name. The database alias name is used to provide the name of the database server as it is known to the network. When a database alias name is defined, the location name is used by the application to reference the server, but the database alias name is used to identify the database server to be accessed. Any fully qualified object names within any SQL statements are not modified and are sent unchanged to the database server.

- **database descriptor (DBD).** An internal representation of a DB2 database definition, which reflects the data definition that is in the DB2 catalog. The objects that are defined in a database descriptor are table spaces, tables, indexes, index spaces, relationships, check constraints, and triggers. A DBD also contains information about accessing tables in the database.

- **database exception status.** An indication that something is wrong with a database. All members of a data sharing group must know and share the exception status of databases.

- **database identifier (DBID).** An internal identifier of the database.

- **database management system (DBMS).** A software system that controls the creation, organization, and modification of a database and the access to the data that is stored within it.

- **database request module (DBRM).** A data set member that is created by the DB2 precompiler and that contains information about SQL statements. DBRMs are used in the bind process.

- **database server.** The target of a request from a local application or an intermediate database server. In the DB2 environment, the database server function is provided by the distributed data facility to access DB2 data from local applications, or from a remote database server that acts as an intermediate database server.

- **data currency.** The state in which data that is retrieved into a host variable in your program is a copy of data in the base table.

- **data definition name (ddname).** The name of a data definition (DD) statement that corresponds to a data control block containing the same name.

- **data dictionary.** A repository of information about an organization’s application programs, databases, logical data models, users, and authorizations. A data dictionary can be manual or automated.

- **data-driven business rules.** Constraints on particular data values that exist as a result of requirements of the business.

- **Data Language/I (DL/I).** The IMS data manipulation language; a common high-level interface between a user application and IMS.

- **data mart.** A small data warehouse that applies to a single department or team. See also data warehouse.

- **data mining.** The process of collecting critical business information from a data warehouse, correlating it, and uncovering associations, patterns, and trends.

- **data partition.** A VSAM data set that is contained within a partitioned table space.

- **data-partitioned secondary index (DPSI).** A secondary index that is partitioned. The index is partitioned according to the underlying data.

- **data sharing.** The ability of two or more DB2 subsystems to directly access and change a single set of data.

- **data sharing group.** A collection of one or more DB2 subsystems that directly access and change the same data while maintaining data integrity.

- **data sharing member.** A DB2 subsystem that is assigned by XCF services to a data sharing group.

- **data source.** A local or remote relational or non-relational data manager that is capable of supporting data access via an ODBC driver that supports the ODBC APIs. In the case of DB2 UDB for z/OS, the data sources are always relational database managers.

- **data space.** In releases prior to DB2 UDB for z/OS, Version 8, a range of up to 2 GB of contiguous virtual storage addresses that a program can directly manipulate. Unlike an address space, a data space can hold only data; it does not contain common areas, system data, or programs.

- **data type.** An attribute of columns, literals, host variables, special registers, and the results of functions and expressions.

- **data warehouse.** A system that provides critical business information to an organization. The data warehouse system cleanses the data for accuracy and currency, and then presents the data to decision makers so that they can interpret and use it effectively and efficiently.

- **date.** A three-part value that designates a day, month, and year.

- **date duration.** A decimal integer that represents a number of years, months, and days.

- **datetime value.** A value of the data type DATE, TIME, or TIMESTAMP.

DBA. Database administrator.
DBCLOB. Double-byte character large object.

DBCS. Double-byte character set.

DBD. Database descriptor.

DBID. Database identifier.

DBMS. Database management system.

DBRM. Database request module.

DB2 catalog. Tables that are maintained by DB2 and contain descriptions of DB2 objects, such as tables, views, and indexes.

DB2 command. An instruction to the DB2 subsystem that a user enters to start or stop DB2, to display information on current users, to start or stop databases, to display information on the status of databases, and so on.

DB2 for VSE & VM. The IBM DB2 relational database management system for the VSE and VM operating systems.

DB2I. DB2 Interactive.

DB2 Interactive (DB2I). The DB2 facility that provides for the execution of SQL statements, DB2 (operator) commands, programmer commands, and utility invocation.

DB2I Kanji Feature. The tape that contains the panels and jobs that allow a site to display DB2I panels in Kanji.

DB2 PM. DB2 Performance Monitor.

DB2 thread. The DB2 structure that describes an application’s connection, traces its progress, processes resource functions, and delimits its accessibility to DB2 resources and services.

DCLGEN. Declarations generator.

DDF. Distributed data facility.

ddbname. Data definition name.

deadlock. Unresolvable contention for the use of a resource, such as a table or an index.

declarations generator (DCLGEN). A subcomponent of DB2 that generates SQL table declarations and COBOL, C, or PL/I data structure declarations that conform to the table. The declarations are generated from DB2 system catalog information. DCLGEN is also a DSN subcommand.

declared temporary table. A table that holds temporary data and is defined with the SQL statement DECLARE GLOBAL TEMPORARY TABLE. Information about declared temporary tables is not stored in the DB2 catalog, so this kind of table is not persistent and can be used only by the application process that issued the DECLARE statement. Contrast with created temporary table. See also temporary table.

default value. A predetermined value, attribute, or option that is assumed when no other is explicitly specified.

defered embedded SQL. SQL statements that are neither fully static nor fully dynamic. Like static statements, they are embedded within an application, but like dynamic statements, they are prepared during the execution of the application.

defered write. The process of asynchronously writing changed data pages to disk.

degree of parallelism. The number of concurrently executed operations that are initiated to process a query.

delete-connected. A table that is a dependent of table $P$ or a dependent of a table to which delete operations from table $P$ cascade.

deflete hole. The location on which a cursor is positioned when a row in a result table is refetched and the row no longer exists on the base table, because another cursor deleted the row between the time the cursor first included the row in the result table and the time the cursor tried to refetch it.

deflete rule. The rule that tells DB2 what to do to a dependent row when a parent row is deleted. For each relationship, the rule might be CASCADE, Restrict, SET NULL, or NO ACTION.

delete trigger. A trigger that is defined with the triggering SQL operation DELETE.

delimited identifier. A sequence of characters that are enclosed within double quotation marks ("), the sequence must consist of a letter followed by zero or more characters, each of which is a letter, digit, or the underscore character (_).

delimiter token. A string constant, a delimited identifier, an operator symbol, or any of the special characters that are shown in DB2 syntax diagrams.

denormalization. A key step in the task of building a physical relational database design. Denormalization is the intentional duplication of columns in multiple tables, and the consequence is increased data redundancy. Denormalization is sometimes necessary to minimize performance problems. Contrast with normalization.

dependent. An object (row, table, or table space) that has at least one parent. The object is also said to be a dependent (row, table, or table space) of its parent. See also parent row, parent table, parent table space.
dependent row • drain lock

dependent row. A row that contains a foreign key that matches the value of a primary key in the parent row.

dependent table. A table that is a dependent in at least one referential constraint.

DES-based authenticator. An authenticator that is generated using the DES algorithm.

descendent. An object that is a dependent of an object or is the dependent of a descendent of an object.

descendent row. A row that is dependent on another row, or a row that is a descendent of a dependent row.

descendent table. A table that is a dependent of another table, or a table that is a descendent of a dependent table.

deterministic function. A user-defined function whose result is dependent on the values of the input arguments. That is, successive invocations with the same input values produce the same answer. Sometimes referred to as a not-variant function. Contrast this with a nondeterministic function (sometimes called a variant function), which might not always produce the same result for the same inputs.

DFP. Data Facility Product (in z/OS).

DFSM. Data Facility Storage Management Subsystem (in z/OS). Also called Storage Management Subsystem (SMS).

DFSMSdss™. The data set services (dss) component of DFSMS (in z/OS).

DFSMSShm. The hierarchical storage manager (hsm) component of DFSMS (in z/OS).

dimension. A data category such as time, products, or markets. The elements of a dimension are referred to as members. Dimensions offer a very concise, intuitive way of organizing and selecting data for retrieval, exploration, and analysis. See also dimension table.

dimension table. The representation of a dimension in a star schema. Each row in a dimension table represents all of the attributes for a particular member of the dimension. See also dimension, star schema, and star join.

directory. The DB2 system database that contains internal objects such as database descriptors and skeleton cursor tables.

# distinct predicate. In SQL, a predicate that ensures that two row values are not equal, and that both row values are not null.

# distinct type. A user-defined data type that is internally represented as an existing type (its source type), but is considered to be a separate and incompatible type for semantic purposes.

distributed data. Data that resides on a DBMS other than the local system.

distributed data facility (DDF). A set of DB2 components through which DB2 communicates with another relational database management system.

Distributed Relational Database Architecture (DRDA). A connection protocol for distributed relational database processing that is used by IBM’s relational database products. DRDA includes protocols for communication between an application and a remote relational database management system, and for communication between relational database management systems. See also DRDA access.

DL/I. Data Language/I.

DNS. Domain name server.

document access definition (DAD). Used to define the indexing scheme for an XML column or the mapping scheme of an XML column. It can be used to enable an XML Extender column of an XML collection, which is XML formatted.

domain. The set of valid values for an attribute.

domain name. The name by which TCP/IP applications refer to a TCP/IP host within a TCP/IP network.

domain name server (DNS). A special TCP/IP network server that manages a distributed directory that is used to map TCP/IP host names to IP addresses.

double-byte character large object (DBCLOB). A sequence of bytes representing double-byte characters where the size of the values can be up to 2 GB. In general, DBCLOB values are used whenever a double-byte character string might exceed the limits of the VARGRAPHIC type.

double-byte character set (DBCS). A set of characters, which are used by national languages such as Japanese and Chinese, that have more symbols than can be represented by a single byte. Each character is 2 bytes in length. Contrast with single-byte character set and multibyte character set.

double-precision floating point number. A 64-bit approximate representation of a real number.

downstream. The set of nodes in the syncpoint tree that is connected to the local DBMS as a participant in the execution of a two-phase commit.

DPSI. Data-partitioned secondary index.

drain. The act of acquiring a locked resource by quiescing access to that object.

drain lock. A lock on a claim class that prevents a claim from occurring.
**DRDA**. Distributed Relational Database Architecture.

**DRDA access**. An open method of accessing distributed data that you can use to connect to another database server to execute packages that were previously bound at the server location. You use the SQL CONNECT statement or an SQL statement with a three-part name to identify the server. Contrast with private protocol access.

**DSN**. (1) The default DB2 subsystem name. (2) The name of the TSO command processor of DB2. (3) The first three characters of DB2 module and macro names.

**duration**. A number that represents an interval of time. See also date duration, labeled duration, and time duration.

**dynamic cursor**. A named control structure that an application program uses to change the size of the result table and the order of its rows after the cursor is opened. Contrast with static cursor.

**dynamic dump**. A dump that is issued during the execution of a program, usually under the control of that program.

**dynamic SQL**. SQL statements that are prepared and executed within an application program while the program is executing. In dynamic SQL, the SQL source is contained in host language variables rather than being coded into the application program. The SQL statement can change several times during the application program’s execution.

**dynamic statement cache pool**. A cache, located above the 2-GB storage line, that holds dynamic statements.

**EA-enabled table space**. A table space or index space that is enabled for extended addressability and that contains individual partitions (or pieces, for LOB table spaces) that are greater than 4 GB.

**EB**. See exabyte.

**EBCDIC**. Extended binary coded decimal interchange code. An encoding scheme that is used to represent character data in the z/OS, VM, VSE, and iSeries™ environments. Contrast with ASCII and Unicode.

**e-business**. The transformation of key business processes through the use of Internet technologies.

**EDM pool**. A pool of main storage that is used for database descriptors, application plans, authorization cache, application packages.

**EID**. Event identifier.

**embedded SQL**. SQL statements that are coded within an application program. See static SQL.

**enclave**. In Language Environment®, an independent collection of routines, one of which is designated as the main routine. An enclave is similar to a program or run unit.

**encoding scheme**. A set of rules to represent character data (ASCII, EBCDIC, or Unicode).

**entity**. A significant object of interest to an organization.

**enumerated list**. A set of DB2 objects that are defined with a LISTDEF utility control statement in which pattern-matching characters (*, %, _ or ?) are not used.

**environment**. A collection of names of logical and physical resources that are used to support the performance of a function.

**environment handle**. In DB2 ODBC, the data object that contains global information regarding the state of the application. An environment handle must be allocated before a connection handle can be allocated. Only one environment handle can be allocated per application.

**EOM**. End of memory.

**EOT**. End of task.

**equijoin**. A join operation in which the join-condition has the form expression = expression.

**error page range**. A range of pages that are considered to be physically damaged. DB2 does not allow users to access any pages that fall within this range.

**escape character**. The symbol that is used to enclose an SQL delimited identifier. The escape character is the double quotation mark ("), except in COBOL applications, where the user assigns the symbol, which is either a double quotation mark or an apostrophe (’).

**ESDS**. Entry sequenced data set.

**ESMT**. External subsystem module table (in IMS).

**EUR**. IBM European Standards.

**exabyte**. For processor, real and virtual storage capacities and channel volume: 1 152 921 504 606 846 976 bytes or 260.

**exception table**. A table that holds rows that violate referential constraints or check constraints that the CHECK DATA utility finds.

**exclusive lock**. A lock that prevents concurrently executing application processes from reading or changing data. Contrast with share lock.

**executable statement**. An SQL statement that can be embedded in an application program, dynamically prepared and executed, or issued interactively.
execution context • foreign key

execution context. In SQLJ, a Java object that can be used to control the execution of SQL statements.

exit routine. A user-written (or IBM-provided default) program that receives control from DB2 to perform specific functions. Exit routines run as extensions of DB2.

expanding conversion. A process that occurs when the length of a converted string is greater than that of the source string. For example, this process occurs when an ASCII mixed-data string that contains DBCS characters is converted to an EBCDIC mixed-data string; the converted string is longer because of the addition of shift codes.

explicit hierarchical locking. Locking that is used to make the parent-child relationship between resources known to IRLM. This kind of locking avoids global locking overhead when no inter-DB2 interest exists on a resource.

exposed name. A correlation name or a table or view name for which a correlation name is not specified. Names that are specified in a FROM clause are exposed or non-exposed.

expression. An operand or a collection of operators and operands that yields a single value.

extended recovery facility (XRF). A facility that minimizes the effect of failures in z/OS, VTAM, the host processor, or high-availability applications during sessions between high-availability applications and designated terminals. This facility provides an alternative subsystem to take over sessions from the failing subsystem.

Extensible Markup Language (XML). A standard metalanguage for defining markup languages that is a subset of Standardized General Markup Language (SGML). The less complex nature of XML makes it easier to write applications that handle document types, to author and manage structured information, and to transmit and share structured information across diverse computing environments.

external function. A function for which the body is written in a programming language that takes scalar argument values and produces a scalar result for each invocation. Contrast with sourced function, built-in function, and SQL function.

external procedure. A user-written application program that can be invoked with the SQL CALL statement, which is written in a programming language. Contrast with SQL procedure.

external routine. A user-defined function or stored procedure that is based on code that is written in an external programming language.

external subsystem module table (ESMT). In IMS, the table that specifies which attachment modules must be loaded.

failed member state. A state of a member of a data sharing group. When a member fails, the XCF permanently records the failed member state. This state usually means that the member’s task, address space, or z/OS system terminated before the state changed from active to quiesced.

fallback. The process of returning to a previous release of DB2 after attempting or completing migration to a current release.

false global lock contention. A contention indication from the coupling facility when multiple lock names are hashed to the same indicator and when no real contention exists.

fan set. A direct physical access path to data, which is provided by an index, hash, or link; a fan set is the means by which the data manager supports the ordering of data.

federated database. The combination of a DB2 Universal Database server (in Linux, UNIX®, and Windows® environments) and multiple data sources to which the server sends queries. In a federated database system, a client application can use a single SQL statement to join data that is distributed across multiple database management systems and can view the data as if it were local.

fetch orientation. The specification of the desired placement of the cursor as part of a FETCH statement (for example, BEFORE, AFTER, NEXT, PRIOR, CURRENT, FIRST, LAST, ABSOLUTE, and RELATIVE).

field procedure. A user-written exit routine that is designed to receive a single value and transform (encode or decode) it in any way the user can specify.

filter factor. A number between zero and one that estimates the proportion of rows in a table for which a predicate is true.

fixed-length string. A character or graphic string whose length is specified and cannot be changed. Contrast with varying-length string.

FlashCopy. A function on the IBM Enterprise Storage Server that can create a point-in-time copy of data while an application is running.

foreign key. A column or set of columns in a dependent table of a constraint relationship. The key must have the same number of columns, with the same descriptions, as the primary key of the parent table.
Each foreign key value must either match a parent key value in the related parent table or be null.

**forest.** An ordered set of subtrees of XML nodes.

**forget.** In a two-phase commit operation, (1) the vote that is sent to the prepare phase when the participant has not modified any data. The forget vote allows a participant to release locks and forget about the logical unit of work. This is also referred to as the read-only vote. (2) The response to the committed request in the second phase of the operation.

**forward log recovery.** The third phase of restart processing during which DB2 processes the log in a forward direction to apply all REDO log records.

**free space.** The total amount of unused space in a page; that is, the space that is not used to store records or control information is free space.

**full outer join.** The result of a join operation that includes the matched rows of both tables that are being joined and preserves the unmatched rows of both tables. See also **join**.

**fullselect.** A subselect, a values-clause, or a number of both that are combined by set operators. **Fullselect** specifies a result table. If **UNION** is not used, the result of the fullselect is the result of the specified subselect.

**fully escaped mapping.** A mapping from an SQL identifier to an XML name when the SQL identifier is a column name.

```
# function. A mapping, which is embodied as a
# program (the function body) that is invocable by means
# of zero or more input values (arguments) to a single
# value (the result). See also **aggregate function** and **scalar function**.
# Functions can be user-defined, built-in, or generated by
# DB2. (See also **built-in function**,** cast function**,** external function**,** sourced function**,** SQL function**,** and **user-defined function**.)

**function definer.** The authorization ID of the owner of the schema of the function that is specified in the CREATE FUNCTION statement.

**function implementer.** The authorization ID of the owner of the function program and function package.

**function package.** A package that results from binding the DBRM for a function program.

**function package owner.** The authorization ID of the user who binds the function program’s DBRM into a function package.

**function resolution.** The process, internal to the DBMS, by which a function invocation is bound to a particular function instance. This process uses the function name, the data types of the arguments, and a list of the applicable schema names (called the **SQL path**) to make the selection. This process is sometimes called **function selection**.

**function selection.** See **function resolution**.

**function signature.** The logical concatenation of a fully qualified function name with the data types of all of its parameters.

**G**

**GB.** Gigabyte (1 073 741 824 bytes).

**GBP.** Group buffer pool.

**GBP-dependent.** The status of a page set or page set partition that is dependent on the group buffer pool. Either read/write interest is active among DB2 subsystems for this page set, or the page set has changed pages in the group buffer pool that have not yet been cast out to disk.

**generalized trace facility (GTF).** A z/OS service program that records significant system events such as I/O interrupts, SVC interrupts, program interrupts, or external interrupts.

**generic resource name.** A name that VTAM uses to represent several application programs that provide the same function in order to handle session distribution and balancing in a Sysplex environment.

**getpage.** An operation in which DB2 accesses a data page.

**global lock.** A lock that provides concurrency control within and among DB2 subsystems. The scope of the lock is across all DB2 subsystems of a data sharing group.

**global lock contention.** Conflicts on locking requests between different DB2 members of a data sharing group when those members are trying to serialize shared resources.

**governor.** See **resource limit facility**.

**graphic string.** A sequence of DBCS characters.

**gross lock.** The **shared**, **update**, or **exclusive** mode locks on a table, partition, or table space.

**group buffer pool (GBP).** A coupling facility cache structure that is used by a data sharing group to cache data and to ensure that the data is consistent for all members.

**group buffer pool duplexing.** The ability to write data to two instances of a group buffer pool structure: a **primary group buffer pool** and a **secondary group buffer pool**.
**group level • image copy**

*pool.* z/OS publications refer to these instances as the "old" (for primary) and "new" (for secondary) structures.

**group level.** The release level of a data sharing group, which is established when the first member migrates to a new release.

**group name.** The z/OS XCF identifier for a data sharing group.

**group restart.** A restart of at least one member of a data sharing group after the loss of either locks or the shared communications area.

**GTF.** Generalized trace facility.

**H**

**handle.** In DB2 ODBC, a variable that refers to a data structure and associated resources. See also statement handle, connection handle, and environment handle.

**help panel.** A screen of information that presents tutorial text to assist a user at the workstation or terminal.

**heuristic damage.** The inconsistency in data between one or more participants that results when a heuristic decision to resolve an indoubt LUW at one or more participants differs from the decision that is recorded at the coordinator.

**heuristic decision.** A decision that forces indoubt resolution at a participant by means other than automatic resynchronization between coordinator and participant.

**hole.** A row of the result table that cannot be accessed because of a delete or an update that has been performed on the row. See also delete hole and update hole.

**home address space.** The area of storage that z/OS currently recognizes as dispatched.

**host.** The set of programs and resources that are available on a given TCP/IP instance.

**host expression.** A Java variable or expression that is referenced by SQL clauses in an SQLJ application program.

**host identifier.** A name that is declared in the host program.

**host language.** A programming language in which you can embed SQL statements.

**host program.** An application program that is written in a host language and that contains embedded SQL statements.

**host structure.** In an application program, a structure that is referenced by embedded SQL statements.

**host variable.** In an application program, an application variable that is referenced by embedded SQL statements.

**host variable array.** An array of elements, each of which corresponds to a value for a column. The dimension of the array determines the maximum number of rows for which the array can be used.

**HSM.** Hierarchical storage manager.

**HTML.** Hypertext Markup Language, a standard method for presenting Web data to users.

**HTTP.** Hypertext Transfer Protocol, a communication protocol that the Web uses.

**I**

**ICF.** Integrated catalog facility.

**IDCAMS.** An IBM program that is used to process access method services commands. It can be invoked as a job or jobstep, from a TSO terminal, or from within a user’s application program.

**IDCAMS LISTCAT.** A facility for obtaining information that is contained in the access method services catalog.

**identify.** A request that an attachment service program in an address space that is separate from DB2 issues through the z/OS subsystem interface to inform DB2 of its existence and to initiate the process of becoming connected to DB2.

**identity column.** A column that provides a way for DB2 to automatically generate a numeric value for each row. The generated values are unique if cycling is not used. Identity columns are defined with the AS IDENTITITY clause. Uniqueness of values can be ensured by defining a unique index that contains only the identity column. A table can have no more than one identity column.

**IFCID.** Instrumentation facility component identifier.

**IFI.** Instrumentation facility interface.

**IFI call.** An invocation of the instrumentation facility interface (IFI) by means of one of its defined functions.

**IFP.** IMS Fast Path.

**image copy.** An exact reproduction of all or part of a table space. DB2 provides utility programs to make full image copies (to copy the entire table space) or incremental image copies (to copy only those pages that have been modified since the last image copy).
implied forget. In the presumed-abort protocol, an implied response of forget to the second-phase committed request from the coordinator. The response is implied when the participant responds to any subsequent request from the coordinator.

IMS. Information Management System.

IMS attachment facility. A DB2 subcomponent that uses z/OS subsystem interface (SSI) protocols and cross-memory linkage to process requests from IMS to DB2 and to coordinate resource commitment.

IMS DB. Information Management System Database.

IMS TM. Information Management System Transaction Manager.

in-abort. A status of a unit of recovery. If DB2 fails after a unit of recovery begins to be rolled back, but before the process is completed, DB2 continues to back out the changes during restart.

in-commit. A status of a unit of recovery. If DB2 fails after beginning its phase 2 commit processing, it "knows," when restarted, that changes made to data are consistent. Such units of recovery are termed in-commit.

independent. An object (row, table, or table space) that is neither a parent nor a dependent of another object.

index. A set of pointers that are logically ordered by the values of a key. Indexes can provide faster access to data and can enforce uniqueness on the rows in a table.

index-controlled partitioning. A type of partitioning in which partition boundaries for a partitioned table are controlled by values that are specified on the CREATE INDEX statement. Partition limits are saved in the LIMITKEY column of the SYSIBM.SYSINDEXPART catalog table.

index key. The set of columns in a table that is used to determine the order of index entries.

index partition. A VSAM data set that is contained within a partitioning index space.

index space. A page set that is used to store the entries of one index.

indicator column. A 4-byte value that is stored in a base table in place of a LOB column.

indicator variable. A variable that is used to represent the null value in an application program. If the value for the selected column is null, a negative value is placed in the indicator variable.

indoubt. A status of a unit of recovery. If DB2 fails after it has finished its phase 1 commit processing and before it has started phase 2, only the commit coordinator knows if an individual unit of recovery is to be committed or rolled back. At emergency restart, if DB2 lacks the information it needs to make this decision, the status of the unit of recovery is indoubt until DB2 obtains this information from the coordinator. More than one unit of recovery can be indoubt at restart.

indoubt resolution. The process of resolving the status of an indoubt logical unit of work to either the committed or the rollback state.

inflight. A status of a unit of recovery. If DB2 fails before its unit of recovery completes phase 1 of the commit process, it merely backs out the updates of its unit of recovery at restart. These units of recovery are termed inflight.

inheritance. The passing downstream of class resources or attributes from a parent class in the class hierarchy to a child class.

initialization file. For DB2 ODBC applications, a file containing values that can be set to adjust the performance of the database manager.

inline copy. A copy that is produced by the LOAD or REORG utility. The data set that the inline copy produces is logically equivalent to a full image copy that is produced by running the COPY utility with read-only access (SHLEVEL REFERENCE).

inner join. The result of a join operation that includes only the matched rows of both tables that are being joined. See also join.

inoperative package. A package that cannot be used because one or more user-defined functions or procedures that the package depends on were dropped. Such a package must be explicitly rebound. Contrast with invalid package.

insensitive cursor. A cursor that is not sensitive to inserts, updates, or deletes that are made to the underlying rows of a result table after the result table has been materialized.

insert trigger. A trigger that is defined with the triggering SQL operation INSERT.

install. The process of preparing a DB2 subsystem to operate as a z/OS subsystem.

installation verification scenario. A sequence of operations that exercises the main DB2 functions and tests whether DB2 was correctly installed.

instrumentation facility component identifier (IFCID). A value that names and identifies a trace record of an event that can be traced. As a parameter on the START TRACe and MODIFY TRACe commands, it specifies that the corresponding event is to be traced.
instrumentation facility interface (IFI) • Kerberos ticket

instrumentation facility interface (IFI). A programming interface that enables programs to obtain online trace data about DB2, to submit DB2 commands, and to pass data to DB2.

Interactive System Productivity Facility (ISPF). An IBM licensed program that provides interactive dialog services in a z/OS environment.

inter-DB2 R/W interest. A property of data in a table space, index, or partition that has been opened by more than one member of a data sharing group and that has been opened for writing by at least one of those members.

intermediate database server. The target of a request from a local application or a remote application requester that is forwarded to another database server. In the DB2 environment, the remote request is forwarded transparently to another database server if the object that is referenced by a three-part name does not reference the local location.

internationalization. The support for an encoding scheme that is able to represent the code points of characters from many different geographies and languages. To support all geographies, the Unicode standard requires more than 1 byte to represent a single character. See also Unicode.

internal resource lock manager (IRLM). A z/OS subsystem that DB2 uses to control communication and database locking.

International Organization for Standardization. An international body charged with creating standards to facilitate the exchange of goods and services as well as cooperation in intellectual, scientific, technological, and economic activity.

invalid package. A package that depends on an object (other than a user-defined function) that is dropped. Such a package is implicitly rebound on invocation. Contrast with inoperative package.

invariant character set. (1) A character set, such as the syntactic character set, whose code point assignments do not change from code page to code page. (2) A minimum set of characters that is available as part of all character sets.

IP address. A 4-byte value that uniquely identifies a TCP/IP host.

IRLM. Internal resource lock manager.

ISO. International Organization for Standardization.

isolation level. The degree to which a unit of work is isolated from the updating operations of other units of work. See also cursor stability, read stability, repeatable read, and uncommitted read.

ISPF. Interactive System Productivity Facility.

ISPF/PDF. Interactive System Productivity Facility/Program Development Facility.

iterator. In SQLJ, an object that contains the result set of a query. An iterator is equivalent to a cursor in other host languages.

iterator declaration clause. In SQLJ, a statement that generates an iterator declaration class. An iterator is an object of an iterator declaration class.

J

Japanese Industrial Standard. An encoding scheme that is used to process Japanese characters.

JAR. Java Archive.

Java Archive (JAR). A file format that is used for aggregating many files into a single file.

JCL. Job control language.

JDBC. A Sun Microsystems database application programming interface (API) for Java that allows programs to access database management systems by using callable SQL. JDBC does not require the use of an SQL preprocessor. In addition, JDBC provides an architecture that lets users add modules called database drivers, which link the application to their choice of database management systems at run time.

JES. Job Entry Subsystem.

JIS. Japanese Industrial Standard.

job control language (JCL). A control language that is used to identify a job to an operating system and to describe the job’s requirements.

Job Entry Subsystem (JES). An IBM licensed program that receives jobs into the system and processes all output data that is produced by the jobs.

join. A relational operation that allows retrieval of data from two or more tables based on matching column values. See also equijoin, full outer join, inner join, left outer join, outer join, and right outer join.

K

KB. Kilobyte (1024 bytes).

Kerberos. A network authentication protocol that is designed to provide strong authentication for client/server applications by using secret-key cryptography.

Kerberos ticket. A transparent application mechanism that transmits the identity of an initiating principal to its target. A simple ticket contains the principal’s
identity, a session key, a timestamp, and other information, which is sealed using the target’s secret key.

**key.** A column or an ordered collection of columns that is identified in the description of a table, index, or referential constraint. The same column can be part of more than one key.

**key-sequenced data set (KSDS).** A VSAM file or data set whose records are loaded in key sequence and controlled by an index.

**keyword.** In SQL, a name that identifies an option that is used in an SQL statement.

**KSDS.** Key-sequenced data set.

**labeled duration.** A number that represents a duration of years, months, days, hours, minutes, seconds, or microseconds.

**large object (LOB).** A sequence of bytes representing bit data, single-byte characters, double-byte characters, or a mixture of single- and double-byte characters. A LOB can be up to 2 GB–1 byte in length. See also BLOB, CLOB, and DBCLOB.

**last agent optimization.** An optimized commit flow for either presumed-nothing or presumed-abort protocols in which the last agent, or final participant, becomes the commit coordinator. This flow saves at least one message.

**latch.** A DB2 internal mechanism for controlling concurrent events or the use of system resources.

**LCID.** Log control interval definition.

**LDS.** Linear data set.

**leaf page.** A page that contains pairs of keys and RIDs and that points to actual data. Contrast with nonleaf page.

**left outer join.** The result of a join operation that includes the matched rows of both tables that are being joined, and that preserves the unmatched rows of the first table. See also join.

**limit key.** The highest value of the index key for a partition.

**linear data set (LDS).** A VSAM data set that contains data but no control information. A linear data set can be accessed as a byte-addressable string in virtual storage.

**linkage editor.** A computer program for creating load modules from one or more object modules or load modules by resolving cross references among the modules and, if necessary, adjusting addresses.

**link-edit.** The action of creating a loadable computer program using a linkage editor.

**list.** A type of object, which DB2 utilities can process, that identifies multiple table spaces, multiple index spaces, or both. A list is defined with the LISTDEF utility control statement.

**list structure.** A coupling facility structure that lets data be shared and manipulated as elements of a queue.

**LLE.** Load list element.

**L-lock.** Logical lock.

**load list element.** A z/OS control block that controls the loading and deleting of a particular load module based on entry point names.

**load module.** A program unit that is suitable for loading into main storage for execution. The output of a linkage editor.

**LOB.** Large object.

**LOB locator.** A mechanism that allows an application program to manipulate a large object value in the database system. A LOB locator is a fullword integer value that represents a single LOB value. An application program retrieves a LOB locator into a host variable and can then apply SQL operations to the associated LOB value using the locator.

**LOB lock.** A lock on a LOB value.

**LOB table space.** A table space in an auxiliary table that contains all the data for a particular LOB column in the related base table.

**local.** A way of referring to any object that the local DB2 subsystem maintains. A local table, for example, is a table that is maintained by the local DB2 subsystem. Contrast with remote.

**locale.** The definition of a subset of a user’s environment that combines a CCSID and characters that are defined for a specific language and country.

**local lock.** A lock that provides intra-DB2 concurrency control, but not inter-DB2 concurrency control; that is, its scope is a single DB2.

**local subsystem.** The unique relational DBMS to which the user or application program is directly connected (in the case of DB2, by one of the DB2 attachment facilities).

**location.** The unique name of a database server. An application uses the location name to access a DB2
**location alias • LU**

| database server. A database alias can be used to override the location name when accessing a remote server. |
| location alias. Another name by which a database server identifies itself in the network. Applications can use this name to access a DB2 database server. |
| lock. A means of controlling concurrent events or access to data. DB2 locking is performed by the IRLM. |
| lock duration. The interval over which a DB2 lock is held. |
| lock escalation. The promotion of a lock from a row, page, or LOB lock to a table space lock because the number of page locks that are concurrently held on a given resource exceeds a preset limit. |
| locking. The process by which the integrity of data is ensured. Locking prevents concurrent users from accessing inconsistent data. |
| lock mode. A representation for the type of access that concurrently running programs can have to a resource that a DB2 lock is holding. |
| lock object. The resource that is controlled by a DB2 lock. |
| lock promotion. The process of changing the size or mode of a DB2 lock to a higher, more restrictive level. |
| lock size. The amount of data that is controlled by a DB2 lock on table data; the value can be a row, a page, a LOB, a partition, a table, or a table space. |
| lock structure. A coupling facility data structure that is composed of a series of lock entries to support shared and exclusive locking for logical resources. |
| log. A collection of records that describe the events that occur during DB2 execution and that indicate their sequence. The information thus recorded is used for recovery in the event of a failure during DB2 execution. |
| log control interval definition. A suffix of the physical log record that tells how record segments are placed in the physical control interval. |
| logical claim. A claim on a logical partition of a nonpartitioning index. |
| logical data modeling. The process of documenting the comprehensive business information requirements in an accurate and consistent format. Data modeling is the first task of designing a database. |
| logical drain. A drain on a logical partition of a nonpartitioning index. |
| logical index partition. The set of all keys that reference the same data partition. |

**logical lock (L-lock).** The lock type that transactions use to control intra- and inter-DB2 data concurrency between transactions. Contrast with *physical lock (P-lock).*

**logically complete.** A state in which the concurrent copy process is finished with the initialization of the target objects that are being copied. The target objects are available for update.

**logical page list (LPL).** A list of pages that are in error and that cannot be referenced by applications until the pages are recovered. The page is in *logical error* because the actual media (coupling facility or disk) might not contain any errors. Usually a connection to the media has been lost.

**logical partition.** A set of key or RID pairs in a nonpartitioning index that are associated with a particular partition.

**logical recovery pending (LRECP).** The state in which the data and the index keys that reference the data are inconsistent.

**logical unit (LU).** An access point through which an application program accesses the SNA network in order to communicate with another application program.

**logical unit of work (LUW).** The processing that a program performs between synchronization points.

**logical unit of work identifier (LUWID).** A name that uniquely identifies a thread within a network. This name consists of a fully-qualified LU network name, an LUW instance number, and an LUW sequence number.

**log initialization.** The first phase of restart processing during which DB2 attempts to locate the current end of the log.

**log record header (LRH).** A prefix, in every logical record, that contains control information.

**log record sequence number (LRSN).** A unique identifier for a log record that is associated with a data sharing member. DB2 uses the LRSN for recovery in the data sharing environment.

**log truncation.** A process by which an explicit starting RBA is established. This RBA is the point at which the next byte of log data is to be written.

**LPL.** Logical page list.

**LRECP.** Logical recovery pending.

**LRH.** Log record header.

**LRSN.** Log record sequence number.

**LU.** Logical unit.
**LU name.** Logical unit name, which is the name by which VTAM refers to a node in a network. Contrast with location name.

**LUW.** Logical unit of work.

**LUWID.** Logical unit of work identifier.

**M**

**mapping table.** A table that the REORG utility uses to map the associations of the RIDs of data records in the original copy and in the shadow copy. This table is created by the user.

**mass delete.** The deletion of all rows of a table.

**master terminal.** The IMS logical terminal that has complete control of IMS resources during online operations.

**master terminal operator (MTO).** See master terminal.

**materialize.** (1) The process of putting rows from a view or nested table expression into a work file for additional processing by a query.

(2) The placement of a LOB value into contiguous storage. Because LOB values can be very large, DB2 avoids materializing LOB data until doing so becomes absolutely necessary.

**materialized query table.** A table that is used to contain information that is derived and can be summarized from one or more source tables.

**MB.** Megabyte (1 048 576 bytes).

**MBCS.** Multibyte character set. UTF-8 is an example of an MBCS. Characters in UTF-8 can range from 1 to 4 bytes in DB2.

**member name.** The z/OS XCF identifier for a particular DB2 subsystem in a data sharing group.

**menu.** A displayed list of available functions for selection by the operator. A menu is sometimes called a menu panel.

**metalanguage.** A language that is used to create other specialized languages.

**migration.** The process of converting a subsystem with a previous release of DB2 to an updated or current release. In this process, you can acquire the functions of the updated or current release without losing the data that you created on the previous release.

**mixed data string.** A character string that can contain both single-byte and double-byte characters.

**MLPA.** Modified link pack area.

**MODEENT.** A VTAM macro instruction that associates a logon mode name with a set of parameters representing session protocols. A set of MODEENT macro instructions defines a logon mode table.

**modeling database.** A DB2 database that you create on your workstation that you use to model a DB2 UDB for z/OS subsystem, which can then be evaluated by the Index Advisor.

**mode name.** A VTAM name for the collection of physical and logical characteristics and attributes of a session.

**modify locks.** An L-lock or P-lock with a MODIFY attribute. A list of these active locks is kept at all times in the coupling facility lock structure. If the requesting DB2 subsystem fails, that DB2 subsystem’s modify locks are converted to retained locks.

**MPP.** Message processing program (in IMS).

**MTO.** Master terminal operator.

**multibyte character set (MBCS).** A character set that represents single characters with more than a single byte. Contrast with single-byte character set and double-byte character set. See also Unicode.

**multidimensional analysis.** The process of assessing and evaluating an enterprise on more than one level.

**Multiple Virtual Storage.** An element of the z/OS operating system. This element is also called the Base Control Program (BCP).

**multisite update.** Distributed relational database processing in which data is updated in more than one location within a single unit of work.

**multithreading.** Multiple TCBs that are executing one copy of DB2 ODBC code concurrently (sharing a processor) or in parallel (on separate central processors).

**must-complete.** A state during DB2 processing in which the entire operation must be completed to maintain data integrity.

**mutex.** Pthread mutual exclusion; a lock. A Pthread mutex variable is used as a locking mechanism to allow serialization of critical sections of code by temporarily blocking the execution of all but one thread.

**MVS.** See Multiple Virtual Storage.

**N**

**negotiable lock.** A lock whose mode can be downgraded, by agreement among contending users, to be compatible to all. A physical lock is an example of a negotiable lock.
nested table expression.  A fullselect in a FROM clause (surrounded by parentheses).

network identifier (NID).  The network ID that is assigned by IMS or CICS, or if the connection type is RRSNF, the RRS unit of recovery ID (URID).

NID.  Network identifier.

nonleaf page.  A page that contains keys and page numbers of other pages in the index (either leaf or nonleaf pages). Nonleaf pages never point to actual data.

nonpartitioned index.  An index that is not physically partitioned. Both partitioning indexes and secondary indexes can be nonpartitioned.

nonscrollable cursor.  A cursor that can be moved only in a forward direction. Nonscrollable cursors are sometimes called forward-only cursors or serial cursors.

normalization.  A key step in the task of building a logical relational database design. Normalization helps you avoid redundancies and inconsistencies in your data. An entity is normalized if it meets a set of constraints for a particular normal form (first normal form, second normal form, and so on). Contrast with denormalization.

nondeterministic function.  A user-defined function whose result is not solely dependent on the values of the input arguments. That is, successive invocations with the same argument values can produce a different answer. this type of function is sometimes called a variant function. Contrast this with a deterministic function (sometimes called a not-variant function), which always produces the same result for the same inputs.

not-variant function.  See deterministic function.

NPSI.  See nonpartitioned secondary index.

NRE.  Network recovery element.

NUL.  The null character (‘\0’), which is represented by the value X’00’. In C, this character denotes the end of a string.

null.  A special value that indicates the absence of information.

NULLIF.  A scalar function that evaluates two passed expressions, returning either NULL if the arguments are equal or the value of the first argument if they are not.

null-terminated host variable.  A varying-length host variable in which the end of the data is indicated by a null terminator.

null terminator.  In C, the value that indicates the end of a string. For EBCDIC, ASCII, and Unicode UTF-8 strings, the null terminator is a single-byte value (X’00’).

For Unicode UCS-2 (wide) strings, the null terminator is a double-byte value (X’0000’).

OASN (origin application schedule number).  In IMS, a 4-byte number that is assigned sequentially to each IMS schedule since the last cold start of IMS. The OASN is used as an identifier for a unit of work. In an 8-byte format, the first 4 bytes contain the schedule number and the last 4 bytes contain the number of IMS sync points (commit points) during the current schedule. The OASN is part of the NID for an IMS connection.

ODBC.  Open Database Connectivity.

ODBC driver.  A dynamically-linked library (DLL) that implements ODBC function calls and interacts with a data source.

OBID.  Data object identifier.

Open Database Connectivity (ODBC).  A Microsoft® database application programming interface (API) for C that allows access to database management systems by using callable SQL. ODBC does not require the use of an SQL preprocessor. In addition, ODBC provides an architecture that lets users add modules called database drivers, which link the application to their choice of database management systems at run time. This means that applications no longer need to be directly linked to the modules of all the database management systems that are supported.

ordinary identifier.  An uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character. An ordinary identifier must not be a reserved word.

ordinary token.  A numeric constant, an ordinary identifier, a host identifier, or a keyword.

originating task.  In a parallel group, the primary agent that receives data from other execution units (referred to as parallel tasks) that are executing portions of the query in parallel.


outer join.  The result of a join operation that includes the matched rows of both tables that are being joined and preserves some or all of the unmatched rows of the tables that are being joined. See also join.

overloaded function.  A function name for which multiple function instances exist.
package • partitioned table space

package. An object containing a set of SQL statements that have been statically bound and that is available for processing. A package is sometimes also called an application package.

package list. An ordered list of package names that may be used to extend an application plan.

package name. The name of an object that is created by a BIND PACKAGE or REBIND PACKAGE command. The object is a bound version of a database request module (DBRM). The name consists of a location name, a collection ID, a package ID, and a version ID.

page. A unit of storage within a table space (4 KB, 8 KB, 16 KB, or 32 KB) or index space (4 KB). In a table space, a page contains one or more rows of a table. In a LOB table space, a LOB value can span more than one page, but no more than one LOB value is stored on a page.

page set. Another way to refer to a table space or index space. Each page set consists of a collection of VSAM data sets.

page set recovery pending (PSRCP). A restrictive state of an index space. In this case, the entire page set must be recovered. Recovery of a logical part is prohibited.

panel. A predefined display image that defines the locations and characteristics of display fields on a display surface (for example, a menu panel).

parallel complex. A cluster of machines that work together to handle multiple transactions and applications.

parallel group. A set of consecutive operations that execute in parallel and that have the same number of parallel tasks.

parallel I/O processing. A form of I/O processing in which DB2 initiates multiple concurrent requests for a single user query and performs I/O processing concurrently (in parallel) on multiple data partitions.

parallelism assistant. In Sysplex query parallelism, a DB2 subsystem that helps to process parts of a parallel query that originates on another DB2 subsystem in the data sharing group.

parallelism coordinator. In Sysplex query parallelism, the DB2 subsystem from which the parallel query originates.

Parallel Sysplex. A set of z/OS systems that communicate and cooperate with each other through certain multisystem hardware components and software services to process customer workloads.

parallel task. The execution unit that is dynamically created to process a query in parallel. A parallel task is implemented by a z/OS service request block.

parameter marker. A question mark (?) that appears in a statement string of a dynamic SQL statement. The question mark can appear where a host variable could appear if the statement string were a static SQL statement.

parameter-name. An SQL identifier that designates a parameter in an SQL procedure or an SQL function.

parent key. A primary key or unique key in the parent table of a referential constraint. The values of a parent key determine the valid values of the foreign key in the referential constraint.

parent lock. For explicit hierarchical locking, a lock that is held on a resource that might have child locks that are lower in the hierarchy. A parent lock is usually the table space lock or the partition intent lock. See also child lock.

parent row. A row whose primary key value is the foreign key value of a dependent row.

parent table. A table whose primary key is referenced by the foreign key of a dependent table.

parent table space. A table space that contains a parent table. A table space containing a dependent of that table is a dependent table space.

participant. An entity other than the commit coordinator that takes part in the commit process. The term participant is synonymous with agent in SNA.

partition. A portion of a page set. Each partition corresponds to a single, independently extendable data set. Partitions can be extended to a maximum size of 1, 2, or 4 GB, depending on the number of partitions in the partitioned page set. All partitions of a given page set have the same maximum size.

partitioned data set (PDS). A data set in disk storage that is divided into partitions, which are called members. Each partition can contain a program, part of a program, or data. The term partitioned data set is synonymous with program library.

partitioned index. An index that is physically partitioned. Both partitioning indexes and secondary indexes can be partitioned.

partitioned page set. A partitioned table space or an index space. Header pages, space map pages, data pages, and index pages reference data only within the scope of the partition.

partitioned table space. A table space that is subdivided into parts (based on index key range), each of which can be processed independently by utilities.
partitioning index • primary authorization ID

partitioning index. An index in which the leftmost columns are the partitioning columns of the table. The index can be partitioned or nonpartitioned.

partition pruning. The removal from consideration of inapplicable partitions through setting up predicates in a query on a partitioned table to access only certain partitions to satisfy the query.

partner logical unit. An access point in the SNA network that is connected to the local DB2 subsystem by way of a VTAM conversation.

path. See SQL path.

PCT. Program control table (in CICS).

PDS. Partitioned data set.

piece. A data set of a nonpartitioned page set.

physical claim. A claim on an entire nonpartitioning index.

physical consistency. The state of a page that is not in a partially changed state.

physical drain. A drain on an entire nonpartitioning index.

physical lock (P-lock). A type of lock that DB2 acquires to provide consistency of data that is cached in different DB2 subsystems. Physical locks are used only in data sharing environments. Contrast with logical lock (L-lock).

physical lock contention. Conflicting states of the requesters for a physical lock. See also negotiable lock.

physically complete. The state in which the concurrent copy process is completed and the output data set has been created.

plan. See application plan.

plan allocation. The process of allocating DB2 resources to a plan in preparation for execution.

plan member. The bound copy of a DBRM that is identified in the member clause.

plan name. The name of an application plan.

plan segmentation. The dividing of each plan into sections. When a section is needed, it is independently brought into the EDM pool.

P-lock. Physical lock.

PLT. Program list table (in CICS).

point of consistency. A time when all recoverable data that an application accesses is consistent with other data. The term point of consistency is synonymous with sync point or commit point.

policy. See CFRM policy.

Portable Operating System Interface (POSIX). The IEEE operating system interface standard, which defines the Pthread standard of threading. See also Pthread.

POSIX. Portable Operating System Interface.

postponed abort UR. A unit of recovery that was infilt or in-abort, was interrupted by system failure or cancellation, and did not complete backout during restart.

PPT. (1) Processing program table (in CICS). (2) Program properties table (in z/OS).

precision. In SQL, the total number of digits in a decimal number (called the size in the C language). In the C language, the number of digits to the right of the decimal point (called the scale in SQL). The DB2 library uses the SQL terms.

precompilation. A processing of application programs containing SQL statements that takes place before compilation. SQL statements are replaced with statements that are recognized by the host language compiler. Output from this precompilation includes source code that can be submitted to the compiler and the database request module (DBRM) that is input to the bind process.

predicate. An element of a search condition that expresses or implies a comparison operation.

prefix. A code at the beginning of a message or record.

preformat. The process of preparing a VSAM ESDS for DB2 use, by writing specific data patterns.

prepare. The first phase of a two-phase commit process in which all participants are requested to prepare for commit.

prepared SQL statement. A named object that is the executable form of an SQL statement that has been processed by the PREPARE statement.

presumed-abort. An optimization of the presumed-nothing two-phase commit protocol that reduces the number of recovery log records, the duration of state maintenance, and the number of messages between coordinator and participant. The optimization also modifies the indoubt resolution responsibility.

presumed-nothing. The standard two-phase commit protocol that defines coordinator and participant responsibilities, relative to logical unit of work states, recovery logging, and indoubt resolution.

primary authorization ID. The authorization ID that is used to identify the application process to DB2.
primary group buffer pool. For a duplexed group buffer pool, the structure that is used to maintain the coherency of cached data. This structure is used for page registration and cross-validation. The z/OS equivalent is old structure. Compare with secondary group buffer pool.

primary index. An index that enforces the uniqueness of a primary key.

primary key. In a relational database, a unique, nonnull key that is part of the definition of a table. A table cannot be defined as a parent unless it has a unique key or primary key.

principal. An entity that can communicate securely with another entity. In Kerberos, principals are represented as entries in the Kerberos registry database and include users, servers, computers, and others.

principal name. The name by which a principal is known to the DCE security services.

private connection. A communications connection that is specific to DB2.

private protocol access. A method of accessing distributed data by which you can direct a query to another DB2 system. Contrast with DRDA access.

private protocol connection. A DB2 private connection of the application process. See also private connection.

privilege. The capability of performing a specific function, sometimes on a specific object. The types of privileges are:

- explicit privileges, which have names and are held as the result of SQL GRANT and REVOKE statements. For example, the SELECT privilege.
- implicit privileges, which accompany the ownership of an object, such as the privilege to drop a synonym that one owns, or the holding of an authority, such as the privilege of SYSADM authority to terminate any utility job.

privilege set. For the installation SYSADM ID, the set of all possible privileges. For any other authorization ID, the set of all privileges that are recorded for that ID in the DB2 catalog.

process. In DB2, the unit to which DB2 allocates resources and locks. Sometimes called an application process, a process involves the execution of one or more programs. The execution of an SQL statement is always associated with some process. The means of initiating and terminating a process are dependent on the environment.

program. A single, compilable collection of executable statements in a programming language.

program temporary fix (PTF). A solution or bypass of a problem that is diagnosed as a result of a defect in a current unaltered release of a licensed program. An authorized program analysis report (APAR) fix is corrective service for an existing problem. A PTF is preventive service for problems that might be encountered by other users of the product. A PTF is temporary, because a permanent fix is usually not incorporated into the product until its next release.

protected conversation. A VTAM conversation that supports two-phase commit flows.

PSRCP. Page set recovery pending.

PTF. Program temporary fix.

Pthread. The POSIX threading standard model for splitting an application into subtasks. The Pthread standard includes functions for creating threads, terminating threads, synchronizing threads through locking, and other thread control facilities.

Q

QMF™. Query Management Facility.

QSAM. Queued sequential access method.

query. A component of certain SQL statements that specifies a result table.

query block. The part of a query that is represented by one of the FROM clauses. Each FROM clause can have multiple query blocks, depending on DB2’s internal processing of the query.

query CP parallelism. Parallel execution of a single query, which is accomplished by using multiple tasks. See also Sysplex query parallelism.

query I/O parallelism. Parallel access of data, which is accomplished by triggering multiple I/O requests within a single query.

queued sequential access method (QSAM). An extended version of the basic sequential access method (BSAM). When this method is used, a queue of data blocks is formed. Input data blocks await processing, and output data blocks await transfer to auxiliary storage or to an output device.

quiesce point. A point at which data is consistent as a result of running the DB2 QUIESCE utility.

quiesced member state. A state of a member of a data sharing group. An active member becomes quiesced when a STOP DB2 command takes effect without a failure. If the member’s task, address space, or z/OS system fails before the command takes effect, the member state is failed.
R

RACF • referential integrity

RACF. Resource Access Control Facility, which is a component of the z/OS Security Server.

RAMAC. IBM family of enterprise disk storage system products.

RBA. Relative byte address.

RCT. Resource control table (in CICS attachment facility).

RDB. Relational database.

RDBMS. Relational database management system.

RDBNAM. Relational database name.

RDF. Record definition field.

read stability (RS). An isolation level that is similar to repeatable read but does not completely isolate an application process from all other concurrently executing application processes. Under level RS, an application that issues the same query more than once might read additional rows that were inserted and committed by a concurrently executing application process.

rebind. The creation of a new application plan for an application program that has been bound previously. If, for example, you have added an index for a table that your application accesses, you must rebind the application in order to take advantage of that index.

rebuild. The process of reallocating a coupling facility structure. For the shared communications area (SCA) and lock structure, the structure is repopulated; for the group buffer pool, changed pages are usually cast out to disk, and the new structure is populated only with changed pages that were not successfully cast out.

RECFM. Record format.

record. The storage representation of a row or other data.

record identifier (RID). A unique identifier that DB2 uses internally to identify a row of data in a table. Compare with row ID.

record identifier (RID) pool. An area of main storage that is used for sorting record identifiers during list-prefetch processing.

record length. The sum of the length of all the columns in a table, which is the length of the data as it is physically stored in the database. Records can be fixed length or varying length, depending on how the columns are defined. If all columns are fixed-length columns, the record is a fixed-length record. If one or more columns are varying-length columns, the record is a varying-length column.

Recoverable Resource Manager Services attachment facility (RRSAF). A DB2 subcomponent that uses Resource Recovery Services to coordinate resource commitment between DB2 and all other resource managers that also use RRS in a z/OS system.

recovery. The process of rebuilding databases after a system failure.

recovery log. A collection of records that describes the events that occur during DB2 execution and indicates their sequence. The recorded information is used for recovery in the event of a failure during DB2 execution.

recovery manager. (1) A subcomponent that supplies coordination services that control the interaction of DB2 resource managers during commit, abort, checkpoint, and restart processes. The recovery manager also supports the recovery mechanisms of other subsystems (for example, IMS) by acting as a participant in the other subsystem’s process for protecting data that has reached a point of consistency. (2) A coordinator or a participant (or both), in the execution of a two-phase commit, that can access a recovery log that maintains the state of the logical unit of work and names the immediate upstream coordinator and downstream participants.

recovery pending (RECP). A condition that prevents SQL access to a table space that needs to be recovered.

recovery token. An identifier for an element that is used in recovery (for example, NID or URID).

RECP. Recovery pending.

redo. A state of a unit of recovery that indicates that changes are to be reapplied to the disk media to ensure data integrity.

reentrant. Executable code that can reside in storage as one shared copy for all threads. Reentrant code is not self-modifying and provides separate storage areas for each thread. Reentrancy is a compiler and operating system concept, and reentrancy alone is not enough to guarantee logically consistent results when multithreading. See also threadsafe.

referential constraint. The requirement that nonnull values of a designated foreign key are valid only if they equal values of the primary key of a designated table.

referential integrity. The state of a database in which all values of all foreign keys are valid. Maintaining referential integrity requires the enforcement of referential constraints on all operations that change the data in a table on which the referential constraints are defined.
**referential structure**. A set of tables and relationships that includes at least one table and, for every table in the set, all the relationships in which that table participates and all the tables to which it is related.

**refresh age**. The time duration between the current time and the time during which a materialized query table was last refreshed.

**registry**. See registry database.

**registry database**. A database of security information about principals, groups, organizations, accounts, and security policies.

**relational database (RDB)**. A database that can be perceived as a set of tables and manipulated in accordance with the relational model of data.

**relational database management system (RDBMS)**. A collection of hardware and software that organizes and provides access to a relational database.

**relational database name (RDBNAM)**. A unique identifier for an RDBMS within a network. In DB2, this must be the value in the LOCATION column of table SYSIBM.LOCATIONS in the CDB. DB2 publications refer to the name of another RDBMS as a LOCATION value or a location name.

**relationship**. A defined connection between the rows of a table or the rows of two tables. A relationship is the internal representation of a referential constraint.

**relative byte address (RBA)**. The offset of a data record or control interval from the beginning of the storage space that is allocated to the data set or file to which it belongs.

**remigration**. The process of returning to a current release of DB2 following a fallback to a previous release. This procedure constitutes another migration process.

**remote**. Any object that is maintained by a remote DB2 subsystem (that is, by a DB2 subsystem other than the local one). A remote view, for example, is a view that is maintained by a remote DB2 subsystem. Contrast with local.

**remote attach request**. A request by a remote location to attach to the local DB2 subsystem. Specifically, the request that is sent is an SNA Function Management Header 5.

**remote subsystem**. Any relational DBMS, except the local subsystem, with which the user or application can communicate. The subsystem need not be remote in any physical sense, and might even operate on the same processor under the same z/OS system.

**reoptimization**. The DB2 process of reconsidering the access path of an SQL statement at run time; during reoptimization, DB2 uses the values of host variables, parameter markers, or special registers.

**REORG pending (REORP)**. A condition that restricts SQL access and most utility access to an object that must be reorganized.

**REORP**. REORG pending.

**repeatable read (RR)**. The isolation level that provides maximum protection from other executing application programs. When an application program executes with repeatable read protection, rows that the program references cannot be changed by other programs until the program reaches a commit point.

**repeating group**. A situation in which an entity includes multiple attributes that are inherently the same. The presence of a repeating group violates the requirement of first normal form. In an entity that satisfies the requirement of first normal form, each attribute is independent and unique in its meaning and its name. See also normalization.

**replay detection mechanism**. A method that allows a principal to detect whether a request is a valid request from a source that can be trusted or whether an untrustworthy entity has captured information from a previous exchange and is replaying the information exchange to gain access to the principal.

**request commit**. The vote that is submitted to the prepare phase if the participant has modified data and is prepared to commit or roll back.

**requester**. The source of a request to access data at a remote server. In the DB2 environment, the requester function is provided by the distributed data facility.

**resource**. The object of a lock or claim, which could be a table space, an index space, a data partition, an index partition, or a logical partition.

**resource allocation**. The part of plan allocation that deals specifically with the database resources.

**resource control table (RCT)**. A construct of the CICS attachment facility, created by site-provided macro parameters, that defines authorization and access attributes for transactions or transaction groups.

**resource definition online**. A CICS feature that you use to define CICS resources online without assembling tables.

**resource limit facility (RLF)**. A portion of DB2 code that prevents dynamic manipulative SQL statements from exceeding specified time limits. The resource limit facility is sometimes called the governor.

**resource limit specification table (RLST)**. A site-defined table that specifies the limits to be enforced by the resource limit facility.
resource manager • scalar function

resource manager. (1) A function that is responsible for managing a particular resource and that guarantees the consistency of all updates made to recoverable resources within a logical unit of work. The resource that is being managed can be physical (for example, disk or main storage) or logical (for example, a particular type of system service). (2) A participant, in the execution of a two-phase commit, that has recoverable resources that could have been modified. The resource manager has access to a recovery log so that it can commit or roll back the effects of the logical unit of work to the recoverable resources.

restart pending (RESTP). A restrictive state of a page set or partition that indicates that restart (backout) work needs to be performed on the object. All access to the page set or partition is denied except for access by the:

- RECOVER POSTPONED command
- Automatic online backout (which DB2 invokes after restart if the system parameter LBACKOUT=AUTO)

RESTP. Restart pending.

result set. The set of rows that a stored procedure returns to a client application.

result set locator. A 4-byte value that DB2 uses to uniquely identify a query result set that a stored procedure returns.

result table. The set of rows that are specified by a SELECT statement.

retained lock. A MODIFY lock that a DB2 subsystem was holding at the time of a subsystem failure. The lock is retained in the coupling facility lock structure across a DB2 failure.

RID. Record identifier.

RID pool. Record identifier pool.

right outer join. The result of a join operation that includes the matched rows of both tables that are being joined and preserves the unmatched rows of the second join operand. See also join.

RLF. Resource limit facility.

RLST. Resource limit specification table.

RMID. Resource manager identifier.

RO. Read-only access.

rollback. The process of restoring data that was changed by SQL statements to the state at its last commit point. All locks are freed. Contrast with commit.

root page. The index page that is at the highest level (or the beginning point) in an index.

routine. A term that refers to either a user-defined function or a stored procedure.

row. The horizontal component of a table. A row consists of a sequence of values, one for each column of the table.

ROWID. Row identifier.

row identifier (ROWID). A value that uniquely identifies a row. This value is stored with the row and never changes.

row lock. A lock on a single row of data.

rowset. A set of rows for which a cursor position is established.

rowset cursor. A cursor that is defined so that one or more rows can be returned as a rowset for a single FETCH statement, and the cursor is positioned on the set of rows that is fetched.

rowset-positioned access. The ability to retrieve multiple rows from a single FETCH statement.

row-positioned access. The ability to retrieve a single row from a single FETCH statement.

row trigger. A trigger that is defined with the trigger granularity FOR EACH ROW.

RRE. Residual recovery entry (in IMS).

RRSAF. Recoverable Resource Manager Services attachment facility.

RS. Read stability.

RTT. Resource translation table.

RURE. Restart URE.

S

savepoint. A named entity that represents the state of data and schemas at a particular point in time within a unit of work. SQL statements exist to set a savepoint, release a savepoint, and restore data and schemas to the state that the savepoint represents. The restoration of data and schemas to a savepoint is usually referred to as rolling back to a savepoint.

SBCS. Single-byte character set.

SCA. Shared communications area.

# scalar function. An SQL operation that produces a single value from another value and is expressed as a function name, followed by a list of arguments that are enclosed in parentheses. Contrast with aggregate function.
scale. In SQL, the number of digits to the right of the decimal point (called the precision in the C language). The DB2 library uses the SQL definition.

schema. (1) The organization or structure of a database. (2) A logical grouping for user-defined functions, distinct types, triggers, and stored procedures. When an object of one of these types is created, it is assigned to one schema, which is determined by the name of the object. For example, the following statement creates a distinct type T in schema C:

CREATE DISTINCT TYPE C.T ... 

scrollability. The ability to use a cursor to fetch in either a forward or backward direction. The FETCH statement supports multiple fetch orientations to indicate the new position of the cursor. See also fetch orientation.

scrollable cursor. A cursor that can be moved in both a forward and a backward direction.

SDWA. System diagnostic work area.

search condition. A criterion for selecting rows from a table. A search condition consists of one or more predicates.

secondary authorization ID. An authorization ID that has been associated with a primary authorization ID by an authorization exit routine.

secondary group buffer pool. For a duplexed group buffer pool, the structure that is used to back up changed pages that are written to the primary group buffer pool. No page registration or cross-invalidation occurs using the secondary group buffer pool. The z/OS equivalent is new structure.

secondary index. A nonpartitioning index on a partitioned table.

section. The segment of a plan or package that contains the executable structures for a single SQL statement. For most SQL statements, one section in the plan exists for each SQL statement in the source program. However, for cursor-related statements, the DECLARE, OPEN, FETCH, and CLOSE statements reference the same section because they each refer to the SELECT statement that is named in the DECLARE CURSOR statement. SQL statements such as COMMIT, ROLLBACK, and some SET statements do not use a section.

segment. A group of pages that holds rows of a single table. See also segmented table space.

segmented table space. A table space that is divided into equal-sized groups of pages called segments. Segments are assigned to tables so that rows of different tables are never stored in the same segment.

self-referencing constraint. A referential constraint that defines a relationship in which a table is a dependent of itself.


sensitive cursor. A cursor that is sensitive to changes that are made to the database after the result table has been materialized.

sequence. A user-defined object that generates a sequence of numeric values according to user specifications.

sequential data set. A non-DB2 data set whose records are organized on the basis of their successive physical positions, such as on magnetic tape. Several of the DB2 database utilities require sequential data sets.

sequential prefetch. A mechanism that triggers consecutive asynchronous I/O operations. Pages are fetched before they are required, and several pages are read with a single I/O operation.

serial cursor. A cursor that can be moved only in a forward direction.

serialized profile. A Java object that contains SQL statements and descriptions of host variables. The SQLJ translator produces a serialized profile for each connection context.

server. The target of a request from a remote requester. In the DB2 environment, the server function is provided by the distributed data facility, which is used to access DB2 data from remote applications.


service class. An eight-character identifier that is used by the z/OS Workload Manager to associate user performance goals with a particular DDF thread or stored procedure. A service class is also used to classify work on parallelism assistants.

service request block. A unit of work that is scheduled to execute in another address space.

session. A link between two nodes in a VTAM network.

session protocols. The available set of SNA communication requests and responses.

shared communications area (SCA). A coupling facility list structure that a DB2 data sharing group uses for inter-DB2 communication.

share lock. A lock that prevents concurrently executing application processes from changing data, but not from reading data. Contrast with exclusive lock.
shift-in character. A special control character (X'0F') that is used in EBCDIC systems to denote that the subsequent bytes represent SBCS characters. See also shift-out character.

shift-out character. A special control character (X'0E') that is used in EBCDIC systems to denote that the subsequent bytes, up to the next shift-in control character, represent DBCS characters. See also shift-in character.

sign-on. A request that is made on behalf of an individual CICS or IMS application process by an attachment facility to enable DB2 to verify that it is authorized to use DB2 resources.

simple page set. A nonpartitioned page set. A simple page set initially consists of a single data set (page set piece). If and when that data set is extended to 2 GB, another data set is created, and so on, up to a total of 32 data sets. DB2 considers the data sets to be a single contiguous linear address space containing a maximum of 64 GB. Data is stored in the next available location within this address space without regard to any partitioning scheme.

dsmpf. System Management Facilities.

SMP/E. System Modification Program/Extended.

SMS. Storage Management Subsystem.

SNA. Systems Network Architecture.

SNA network. The part of a network that conforms to the formats and protocols of Systems Network Architecture (SNA).

socket. A callable TCP/IP programming interface that TCP/IP network applications use to communicate with remote TCP/IP partners.

sourced function. A function that is implemented by another built-in or user-defined function that is already known to the database manager. This function can be a scalar function or a column (aggregating) function; it returns a single value from a set of values (for example, MAX or AVG). Contrast with built-in function, external function, and SQL function.

source program. A set of host language statements and SQL statements that is processed by an SQL precompiler.

source table. A table that can be a base table, a view, a table expression, or a user-defined table function.

source type. An existing type that DB2 uses to internally represent a distinct type.

space. A sequence of one or more blank characters.

special register. A storage area that DB2 defines for an application process to use for storing information that can be referenced in SQL statements. Examples of special registers are USER and CURRENT DATE.

specific function name. A particular user-defined function that is known to the database manager by its specific name. Many specific user-defined functions can have the same function name. When a user-defined function is defined to the database, every function is assigned a specific name that is unique within its schema. Either the user can provide this name, or a default name is used.

SPUFI. SQL Processor Using File Input.

SQL. Structured Query Language.

SQL authorization ID (SQL ID). The authorization ID that is used for checking dynamic SQL statements in some situations.

SQLCA. SQL communication area.

SQL communication area (SQLCA). A structure that is used to provide an application program with information about the execution of its SQL statements.

SQL connection. An association between an application process and a local or remote application server or database server.

SQLDA. SQL descriptor area.

SQL descriptor area (SQLDA). A structure that describes input variables, output variables, or the columns of a result table.

SQL escape character. The symbol that is used to enclose an SQL delimited identifier. This symbol is the double quotation mark (".). See also escape character.

SQL function. A user-defined function in which the CREATE FUNCTION statement contains the source code. The source code is a single SQL expression that evaluates to a single value. The SQL user-defined function can return only one parameter.

SQL ID. SQL authorization ID.

SQLJ. Structured Query Language (SQL) that is embedded in the Java programming language.
SQL path. An ordered list of schema names that are used in the resolution of unqualified references to user-defined functions, distinct types, and stored procedures. In dynamic SQL, the current path is found in the CURRENT PATH special register. In static SQL, it is defined in the PATH bind option.

SQL procedure. A user-written program that can be invoked with the SQL CALL statement. Contrast with external procedure.

SQL processing conversation. Any conversation that requires access of DB2 data, either through an application or by dynamic query requests.

SQL Processor Using File Input (SPUFI). A facility of the TSO attachment subcomponent that enables the DB2I user to execute SQL statements without embedding them in an application program.

SQL return code. Either SQLCODE or SQLSTATE.

SQL routine. A user-defined function or stored procedure that is based on code that is written in SQL.

SQL statement coprocessor. An alternative to the DB2 precompiler that lets the user process SQL statements at compile time. The user invokes an SQL statement coprocessor by specifying a compiler option.

SQL string delimiter. A symbol that is used to enclose an SQL string constant. The SQL string delimiter is the apostrophe (‘), except in COBOL applications, where the user assigns the symbol, which is either an apostrophe or a double quotation mark (“).

SRB. Service request block.

SSL. Subsystem interface (in z/OS).

SSM. Subsystem member (in IMS).

stand-alone. An attribute of a program that means that it is capable of executing separately from DB2, without using DB2 services.

star join. A method of joining a dimension column of a fact table to the key column of the corresponding dimension table. See also join, dimension, and star schema.

star schema. The combination of a fact table (which contains most of the data) and a number of dimension tables. See also star join, dimension, and dimension table.

statement handle. In DB2 ODBC, the data object that contains information about an SQL statement that is managed by DB2 ODBC. This includes information such as dynamic arguments, bindings for dynamic arguments and columns, cursor information, result values, and status information. Each statement handle is associated with the connection handle.

statement string. For a dynamic SQL statement, the character string form of the statement.

statement trigger. A trigger that is defined with the trigger granularity FOR EACH STATEMENT.

static cursor. A named control structure that does not change the size of the result table or the order of its rows after an application opens the cursor. Contrast with dynamic cursor.

static SQL. SQL statements, embedded within a program, that are prepared during the program preparation process (before the program is executed). After being prepared, the SQL statement does not change (although values of host variables that are specified by the statement might change).

storage group. A named set of disks on which DB2 data can be stored.

stored procedure. A user-written application program that can be invoked through the use of the SQL CALL statement.

string. See character string or graphic string.

strong typing. A process that guarantees that only user-defined functions and operations that are defined on a distinct type can be applied to that type. For example, you cannot directly compare two currency types, such as Canadian dollars and U.S. dollars. But you can provide a user-defined function to convert one currency to the other and then do the comparison.

structure. (1) A name that refers collectively to different types of DB2 objects, such as tables, databases, views, indexes, and table spaces. (2) A construct that uses z/OS to map and manage storage on a coupling facility. See also cache structure, list structure, or lock structure.

Structured Query Language (SQL). A standardized language for defining and manipulating data in a relational database.

structure owner. In relation to group buffer pools, the DB2 member that is responsible for the following activities:
- Coordinating rebuild, checkpoint, and damage assessment processing
- Monitoring the group buffer pool threshold and notifying castout owners when the threshold has been reached

subcomponent. A group of closely related DB2 modules that work together to provide a general function.

subject table. The table for which a trigger is created. When the defined triggering event occurs on this table, the trigger is activated.
subpage. The unit into which a physical index page can be divided.

subquery. A SELECT statement within the WHERE or HAVING clause of another SQL statement; a nested SQL statement.

subselect. That form of a query that does not include an ORDER BY clause, an UPDATE clause, or UNION operators.

substitution character. A unique character that is substituted during character conversion for any characters in the source program that do not have a match in the target coding representation.

subsystem. A distinct instance of a relational database management system (RDBMS).

surrogate pair. A coded representation for a single character that consists of a sequence of two 16-bit code units, in which the first value of the pair is a high-surrogate code unit in the range U+D800 through U+DBFF, and the second value is a low-surrogate code unit in the range U+DC00 through U+DFFF. Surrogate pairs provide an extension mechanism for encoding 917,476 characters without requiring the use of 32-bit characters.

SVC dump. A dump that is issued when a z/OS or a DB2 functional recovery routine detects an error.

sync point. See commit point.

cyclic point. The tree of recovery managers and resource managers that are involved in a logical unit of work, starting with the recovery manager, that make the final commit decision.

synonym. In SQL, an alternative name for a table or view. Synonyms can be used to refer only to objects at the subsystem in which the synonym is defined.

syntactic character set. A set of 81 graphic characters that are registered in the IBM registry as character set 00640. This set was originally recommended to the programming language community to be used for syntactic purposes toward maximizing portability and interchangeability across systems and country boundaries. It is contained in most of the primary registered character sets, with a few exceptions. See also invariant character set.

Sysplex. See Parallel Sysplex.

Sysplex query parallelism. Parallel execution of a single query that is accomplished by using multiple tasks on more than one DB2 subsystem. See also query CP parallelism.

system administrator. The person at a computer installation who designs, controls, and manages the use of the computer system.

system agent. A work request that DB2 creates internally such as prefetch processing, deferred writes, and service tasks.

system conversation. The conversation that two DB2 subsystems must establish to process system messages before any distributed processing can begin.

system diagnostic work area (SDWA). The data that is recorded in a SYS1.LOGREC entry that describes a program or hardware error.

system-directed connection. A connection that a relational DBMS manages by processing SQL statements with three-part names.

System Modification Program/Extended (SMP/E). A z/OS tool for making software changes in programming systems (such as DB2) and for controlling those changes.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information through and controlling the configuration and operation of networks.

SYS1.DUMPxx data set. A data set that contains a system dump (in z/OS).

SYS1.LOGREC. A service aid that contains important information about program and hardware errors (in z/OS).

T

table. A named data object consisting of a specific number of columns and some number of unordered rows. See also base table or temporary table.

table-controlled partitioning. A type of partitioning in which partition boundaries for a partitioned table are controlled by values that are defined in the CREATE TABLE statement. Partition limits are saved in the LIMITKEY_INTERNAL column of the SYSIBM.SYSTABLEPART catalog table.

table function. A function that receives a set of arguments and returns a table to the SQL statement that references the function. A table function can be referenced only in the FROM clause of a subselect.

table locator. A mechanism that allows access to trigger transition tables in the FROM clause of SELECT statements, in the subselect of INSERT statements, or from within user-defined functions. A table locator is a fullword integer value that represents a transition table.

table space. A page set that is used to store the records in one or more tables.
**table space set.** A set of table spaces and partitions that should be recovered together for one of these reasons:

- Each of them contains a table that is a parent or descendent of a table in one of the others.
- The set contains a base table and associated auxiliary tables.

A table space set can contain both types of relationships.

task control block (TCB). A z/OS control block that is used to communicate information about tasks within an address space that are connected to DB2. See also *address space connection.*

TB. Terabyte (1,099,511,627 776 bytes).

TCB. Task control block (in z/OS).

TCP/IP. A network communication protocol that computer systems use to exchange information across telecommunication links.

TCP/IP port. A 2-byte value that identifies an end user or a TCP/IP network application within a TCP/IP host.

template. A DB2 utilities output data set descriptor that is used for dynamic allocation. A template is defined by the TEMPLATE utility control statement.

temporary table. A table that holds temporary data. Temporary tables are useful for holding or sorting intermediate results from queries that contain a large number of rows. The two types of temporary table, which are created by different SQL statements, are the created temporary table and the declared temporary table. Contrast with *result table.* See also created temporary table and declared temporary table.

Terminal Monitor Program (TMP). A program that provides an interface between terminal users and command processors and has access to many system services (in z/OS).

thread. The DB2 structure that describes an application's connection, traces its progress, processes resource functions, and delimits its accessibility to DB2 resources and services. Most DB2 functions execute under a thread structure. See also *allied thread* and *database access thread.*

threadsafe. A characteristic of code that allows multithreading both by providing private storage areas for each thread, and by properly serializing shared (global) storage areas.

three-part name. The full name of a table, view, or alias. It consists of a location name, authorization ID, and an object name, separated by a period.

time. A three-part value that designates a time of day in hours, minutes, and seconds.

time duration. A decimal integer that represents a number of hours, minutes, and seconds.

timeout. Abnormal termination of either the DB2 subsystem or of an application because of the unavailability of resources. Installation specifications are set to determine both the amount of time DB2 is to wait for IRLM services after starting, and the amount of time IRLM is to wait if a resource that an application requests is unavailable. If either of these time specifications is exceeded, a timeout is declared.

Time-Sharing Option (TSO). An option in MVS that provides interactive time sharing from remote terminals.

timestamp. A seven-part value that consists of a date and time. The timestamp is expressed in years, months, days, hours, minutes, seconds, and microseconds.

TMP. Terminal Monitor Program.

to-do. A state of a unit of recovery that indicates that the unit of recovery's changes to recoverable DB2 resources are indoubt and must either be applied to the disk media or backed out, as determined by the commit coordinator.

trace. A DB2 facility that provides the ability to monitor and collect DB2 monitoring, auditing, performance, accounting, statistics, and serviceability (global) data.

transaction lock. A lock that is used to control concurrent execution of SQL statements.

transaction program name. In SNA LU 6.2 conversations, the name of the program at the remote logical unit that is to be the other half of the conversation.

transient XML data type. A data type for XML values that exists only during query processing.

transition table. A temporary table that contains all the affected rows of the subject table in their state before or after the triggering event occurs. Triggered SQL statements in the trigger definition can reference the table of changed rows in the old state or the new state.

transition variable. A variable that contains a column value of the affected row of the subject table in its state before or after the triggering event occurs. Triggered SQL statements in the trigger definition can reference the set of old values or the set of new values.

tree structure. A data structure that represents entities in nodes, with a most one parent node for each node, and with only one root node.
trigger • unique index

trigger. A set of SQL statements that are stored in a DB2 database and executed when a certain event occurs in a DB2 table.

trigger activation. The process that occurs when the trigger event that is defined in a trigger definition is executed. Trigger activation consists of the evaluation of the triggered action condition and conditional execution of the triggered SQL statements.

trigger activation time. An indication in the trigger definition of whether the trigger should be activated before or after the trigger event.

trigger body. The set of SQL statements that is executed when a trigger is activated and its triggered action condition evaluates to true. A trigger body is also called triggered SQL statements.

trigger cascading. The process that occurs when the triggered action of a trigger causes the activation of another trigger.

triggered action. The SQL logic that is performed when a trigger is activated. The triggered action consists of an optional triggered action condition and a set of triggered SQL statements that are executed only if the condition evaluates to true.

triggered action condition. An optional part of the triggered action. This Boolean condition appears as a WHEN clause and specifies a condition that DB2 evaluates to determine if the triggered SQL statements should be executed.

triggered SQL statements. The set of SQL statements that is executed when a trigger is activated and its triggered action condition evaluates to true. Triggered SQL statements are also called the trigger body.

trigger granularity. A characteristic of a trigger, which determines whether the trigger is activated:
• Only once for the triggering SQL statement
• Once for each row that the SQL statement modifies

triggering event. The specified operation in a trigger definition that causes the activation of that trigger. The triggering event is comprised of a triggering operation (INSERT, UPDATE, or DELETE) and a subject table on which the operation is performed.

triggering SQL operation. The SQL operation that causes a trigger to be activated when performed on the subject table.

trigger package. A package that is created when a CREATE TRIGGER statement is executed. The package is executed when the trigger is activated.

TSO. Time-Sharing Option.

TSO attachment facility. A DB2 facility consisting of the DSN command processor and DB2I. Applications that are not written for the CICS or IMS environments can run under the TSO attachment facility.

typed parameter marker. A parameter marker that is specified along with its target data type. It has the general form:
CAST( ? AS data-type)

type 1 indexes. Indexes that were created by a release of DB2 before DB2 Version 4 or that are specified as type 1 indexes in Version 4. Contrast with type 2 indexes. As of Version 8, type 1 indexes are no longer supported.

type 2 indexes. Indexes that are created on a release of DB2 after Version 7 or that are specified as type 2 indexes in Version 4 or later.

U

UCS-2. Universal Character Set, coded in 2 octets, which means that characters are represented in 16-bits per character.

UDF. User-defined function.

UDT. User-defined data type. In DB2 UDB for z/OS, the term distinct type is used instead of user-defined data type. See distinct type.

uncommitted read (UR). The isolation level that allows an application to read uncommitted data.

underlying view. The view on which another view is directly or indirectly defined.

undo. A state of a unit of recovery that indicates that the changes that the unit of recovery made to recoverable DB2 resources must be backed out.

Unicode. A standard that parallels the ISO-10646 standard. Several implementations of the Unicode standard exist, all of which have the ability to represent a large percentage of the characters that are contained in the many scripts that are used throughout the world.

uniform resource locator (URL). A Web address, which offers a way of naming and locating specific items on the Web.

union. An SQL operation that combines the results of two SELECT statements. Unions are often used to merge lists of values that are obtained from several tables.

unique constraint. An SQL rule that no two values in a primary key, or in the key of a unique index, can be the same.

unique index. An index that ensures that no identical key values are stored in a column or a set of columns in a table.
unit of recovery. A recoverable sequence of operations within a single resource manager, such as an instance of DB2. Contrast with unit of work.

unit of recovery identifier (URID). The LOGRBA of the first log record for a unit of recovery. The URID also appears in all subsequent log records for that unit of recovery.

unit of work. A recoverable sequence of operations within an application process. At any time, an application process is a single unit of work, but the life of an application process can involve many units of work as a result of commit or rollback operations. In a multisite update operation, a single unit of work can include several units of recovery. Contrast with unit of recovery.

Universal Unique Identifier (UUID). An identifier that is immutable and unique across time and space (in z/OS).

unlock. The act of releasing an object or system resource that was previously locked and returning it to general availability within DB2.

untyped parameter marker. A parameter marker that is specified without its target data type. It has the form of a single question mark (?)

updatability. The ability of a cursor to perform positioned updates and deletes. The updatability of a cursor can be influenced by the SELECT statement and the cursor sensitivity option that is specified on the DECLARE CURSOR statement.

update hole. The location on which a cursor is positioned when a row in a result table is fetched again and the new values no longer satisfy the search condition. DB2 marks a row in the result table as an update hole when an update to the corresponding row in the database causes that row to no longer qualify for the result table.

update trigger. A trigger that is defined with the triggering SQL operation UPDATE.

upstream. The node in the syncpoint tree that is responsible, in addition to other recovery or resource managers, for coordinating the execution of a two-phase commit.

UR. Uncommitted read.

URE. Unit of recovery element.

URID. Unit of recovery identifier.

URL. Uniform resource locator.

user-defined data type (UDT). See distinct type.

user-defined function (UDF). A function that is defined to DB2 by using the CREATE FUNCTION statement and that can be referenced thereafter in SQL statements. A user-defined function can be an external function, a sourced function, or an SQL function. Contrast with built-in function.

user view. In logical data modeling, a model or representation of critical information that the business requires.

UTF-8. Unicode Transformation Format, 8-bit encoding form, which is designed for ease of use with existing ASCII-based systems. The CCSID value for data in UTF-8 format is 1208. DB2 UDB for z/OS supports UTF-8 in mixed data fields.

UTF-16. Unicode Transformation Format, 16-bit encoding form, which is designed to provide code values for over a million characters and a superset of UCS-2. The CCSID value for data in UTF-16 format is 1200. DB2 UDB for z/OS supports UTF-16 in graphic data fields.

UUID. Universal Unique Identifier.

V

value. The smallest unit of data that is manipulated in SQL.

variable. A data element that specifies a value that can be changed. A COBOL elementary data item is an example of a variable. Contrast with constant.

variant function. See nondeterministic function.

varying-length string. A character or graphic string whose length varies within set limits. Contrast with fixed-length string.

version. A member of a set of similar programs, DBRMs, packages, or LOBs.

A version of a program is the source code that is produced by precompiling the program. The program version is identified by the program name and a timestamp (consistency token).

A version of a DBRM is the DBRM that is produced by precompiling a program. The DBRM version is identified by the same program name and timestamp as a corresponding program version.

A version of a package is the result of binding a DBRM within a particular database system. The package version is identified by the same program name and consistency token as the DBRM.

A version of a LOB is a copy of a LOB value at a point in time. The version number for a LOB is stored in the auxiliary index entry for the LOB.

view. An alternative representation of data from one or more tables. A view can include all or some of the columns that are contained in tables on which it is defined.
view check option. An option that specifies whether every row that is inserted or updated through a view must conform to the definition of that view. A view check option can be specified with the WITH CASCADED CHECK OPTION, WITH CHECK OPTION, or WITH LOCAL CHECK OPTION clauses of the CREATE VIEW statement.

Virtual Storage Access Method (VSAM). An access method for direct or sequential processing of fixed- and varying-length records on disk devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative-record number (in z/OS).

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network (in z/OS).

volatile table. A table for which SQL operations choose index access whenever possible.

VSAM. Virtual Storage Access Method.

VTAM. Virtual Telecommunication Access Method (in z/OS).

W

warm start. The normal DB2 restart process, which involves reading and processing log records so that data that is under the control of DB2 is consistent. Contrast with cold start.

WLM application environment. A z/OS Workload Manager attribute that is associated with one or more stored procedures. The WLM application environment determines the address space in which a given DB2 stored procedure runs.

write to operator (WTO). An optional user-coded service that allows a message to be written to the system console operator informing the operator of errors and unusual system conditions that might need to be corrected (in z/OS).

WTO. Write to operator.

WTOR. Write to operator (WTO) with reply.

X

XCF. See cross-system coupling facility.

XES. See cross-system extended services.

XML. See Extensible Markup Language.
Bibliography

DB2 Universal Database for z/OS Version 8 product information:
- DB2 Administration Guide, SC18-7413
- DB2 Application Programming and SQL Guide, SC18-7415
- DB2 Application Programming Guide and Reference for Java, SC18-7414
- DB2 Codes, GC18-9603
- DB2 Command Reference, SC18-7416
- DB2 Common Criteria Guide, SC18-9672
- DB2 Data Sharing: Planning and Administration, SC18-7417
- DB2 Diagnosis Guide and Reference, LY37-3201
- DB2 Diagnostic Quick Reference Card, LY37-3202
- DB2 Image, Audio, and Video Extenders Administration and Programming, SC18-7429
- DB2 Installation Guide, GC18-7418
- DB2 Licensed Program Specifications, GC18-7420
- DB2 Management Clients Package Program Directory, GI10-8567
- DB2 Messages, GC18-9602
- DB2 ODBC Guide and Reference, SC18-7423
- The Official Introduction to DB2 UDB for z/OS
- DB2 Program Directory, GI10-8566
- DB2 RACF Access Control Module Guide, SC18-7433
- DB2 Reference for Remote DRDA Requesters and Servers, SC18-7424
- DB2 Reference Summary, SX26-3853
- DB2 Release Planning Guide, SC18-7425
- DB2 SQL Reference, SC18-7426
- DB2 Text Extender Administration and Programming, SC18-7430
- DB2 Utility Guide and Reference, SC18-7427
- DB2 What’s New?, GC18-7428
- DB2 XML Extender for z/OS Administration and Programming, SC18-7431

Books and resources about related products:

APL2®
- APL2 Programming Guide, SH21-1072
- APL2 Programming: Language Reference, SH21-1061
- APL2 Programming: Using Structured Query Language (SQL), SH21-1057

BookManager® READ/MVS
- BookManager READ/MVS V1R3: Installation Planning & Customization, SC38-2035

C language: IBM C/C++ for z/OS
- z/OS C/C++ Programming Guide, SC09-4765
- z/OS C/C++ Run-Time Library Reference, SA22-7821

Character Data Representation Architecture
- Character Data Representation Architecture Overview, GC09-2207
- Character Data Representation Architecture Reference and Registry, SC09-2190

CICS Transaction Server for z/OS

The publication order numbers below are for Version 2 Release 2 and Version 2 Release 3 (with the release 2 number listed first).
- CICS Transaction Server for z/OS Information Center, SK3T-6903 or SK3T-6957.
- CICS Transaction Server for z/OS Application Programming Guide, SC34-5993 or SC34-6231
- CICS Transaction Server for z/OS Application Programming Reference, SC34-5994 or SC34-6232
- CICS Transaction Server for z/OS CICS Supplied Transactions, SC34-5992 or SC34-6230
- CICS Transaction Server for z/OS CICS Customization Guide, SC34-5989 or SC34-6227
- CICS Transaction Server for z/OS Data Areas, LY33-6100 or LY33-6103
- CICS Transaction Server for z/OS DB2 Guide, SC34-6014 or SC34-6252
- CICS Transaction Server for z/OS External Interfaces Guide, SC34-6006 or SC34-6244
- CICS Transaction Server for z/OS Installation Guide, GC34-5985 or GC34-6224
- CICS Transaction Server for z/OS Intercommunication Guide, SC34-6005 or SC34-6243
- CICS Transaction Server for z/OS Messages and Codes, GC34-6003 or GC34-6241
- CICS Transaction Server for z/OS Operations and Utilities Guide, SC34-5991 or SC34-6229
• CICS Transaction Server for z/OS Performance Guide, SC34-6009 or SC34-6247
• CICS Transaction Server for z/OS Problem Determination Guide, SC34-6002 or SC34-6239
• CICS Transaction Server for z/OS Release Guide, GC34-5983 or GC34-6218
• CICS Transaction Server for z/OS Resource Definition Guide, SC34-5990 or SC34-6228
• CICS Transaction Server for z/OS System Definition Guide, SC34-5988 or SC34-6226
• CICS Transaction Server for z/OS System Programming Reference, SC34-5595 or SC34-6233

CICS Transaction Server for OS/390
• CICS Transaction Server for OS/390 Application Programming Guide, SC33-1687
• CICS Transaction Server for OS/390 DB2 Guide, SC33-1939
• CICS Transaction Server for OS/390 External Interfaces Guide, SC33-1944
• CICS Transaction Server for OS/390 Resource Definition Guide, SC33-1684

COBOL: IBM COBOL
• IBM COBOL Language Reference, SC27-1408
• IBM COBOL for MVS & VM Programming Guide, SC27-1412

Database Design

DB2 Administration Tool

DB2 Buffer Pool Analyzer for z/OS

DB2 Connect
• IBM DB2 Connect Quick Beginnings for DB2 Connect Enterprise Edition, GC09-4833

• IBM DB2 Connect Quick Beginnings for DB2 Connect Personal Edition, GC09-4834
• IBM DB2 Connect User’s Guide, SC09-4835

DB2 DataPropagator™
• DB2 Universal Database Replication Guide and Reference, SC27-1121

DB2 Performance Expert for z/OS, Version 1

The following books are part of the DB2 Performance Expert library. Some of these books include information about the following tools:
IBM DB2 Performance Expert for z/OS; IBM DB2 Performance Monitor for z/OS; and DB2 Buffer Pool Analyzer for z/OS.
• OMEGAMON Buffer Pool Analyzer User’s Guide, SC18-7972
• OMEGAMON Configuration and Customization, SC18-7973
• OMEGAMON Messages, SC18-7974
• OMEGAMON Monitoring Performance from ISPF, SC18-7975
• OMEGAMON Monitoring Performance from Performance Expert Client, SC18-7976
• OMEGAMON Program Directory, GI10-8549
• OMEGAMON Report Command Reference, SC18-7977
• OMEGAMON Report Reference, SC18-7978
• Using IBM Tivoli OMEGAMON XE on z/OS, SC18-7979

DB2 Query Management Facility (QMF) Version 8.1
• DB2 Query Management Facility: DB2 QMF High Performance Option User’s Guide for TSO/CICS, SC18-7450
• DB2 Query Management Facility: DB2 QMF Messages and Codes, GC18-7447
• DB2 Query Management Facility: DB2 QMF Reference, SC18-7446
• DB2 Query Management Facility: Developing DB2 QMF Applications, SC18-7651
• DB2 Query Management Facility: Getting Started with DB2 QMF for Windows and DB2 QMF for WebSphere, SC18-7449
• DB2 Query Management Facility: Getting Started with DB2 QMF Query Miner, GC18-7451
• DB2 Query Management Facility: Installing and Managing DB2 QMF for TSO/CICS, GC18-7444
• DB2 Query Management Facility: Installing and Managing DB2 QMF for Windows and DB2 QMF for WebSphere, GC18-7448
• DB2 Query Management Facility: Introducing DB2 QMF, GC18-7443
• DB2 Query Management Facility: Using DB2 QMF, SC18-7445
• DB2 Query Management Facility: DB2 QMF Visionary Developer’s Guide, SC18-9093
• DB2 Query Management Facility: DB2 QMF Visionary Getting Started Guide, GC18-9092

DB2 Redbooks™

For access to all IBM Redbooks about DB2, see the IBM Redbooks Web page at www.ibm.com/redbooks

DB2 Server for VSE & VM
• DB2 Server for VM: DBS Utility, SC09-2983

DB2 Universal Database Cross-Platform information

DB2 Universal Database for iSeries

The following books are available at www.ibm.com/iseries/infocenter
• DB2 Universal Database for iSeries Performance and Query Optimization
• DB2 Universal Database for iSeries Database Programming
• DB2 Universal Database for iSeries SQL Programming Concepts
• DB2 Universal Database for iSeries SQL Programming with Host Languages
• DB2 Universal Database for iSeries SQL Reference
• DB2 Universal Database for iSeries Distributed Data Management
• DB2 Universal Database for iSeries Distributed Database Programming

DB2 Universal Database for Linux, UNIX, and Windows:
• DB2 Universal Database Administration Guide: Planning, SC09-4822
• DB2 Universal Database Administration Guide: Implementation, SC09-4820
• DB2 Universal Database Administration Guide: Performance, SC09-4821
• DB2 Universal Database Administrative API Reference, SC09-4824
• DB2 Universal Database Application Development Guide: Building and Running Applications, SC09-4825

• DB2 Universal Database Call Level Interface Guide and Reference, Volumes 1 and 2, SC09-4849 and SC09-4850
• DB2 Universal Database Command Reference, SC09-4828
• DB2 Universal Database SQL Reference Volume 1, SC09-4844
• DB2 Universal Database SQL Reference Volume 2, SC09-4845

Device Support Facilities
• Device Support Facilities User’s Guide and Reference, GC35-0033

DFSMS

These books provide information about a variety of components of DFSMS, including z/OS DFSMS, z/OS DFSMSdfp™, z/OS DFSMSdss, z/OS DFSMShsm, and z/OS DFP.
• z/OS DFSMS Access Method Services for Catalogs, SC26-7394
• z/OS DFSMSdss Storage Administration Guide, SC35-0423
• z/OS DFSMSdss Storage Administration Reference, SC35-0424
• z/OS DFSMShsm Managing Your Own Data, SC35-0420
• z/OS DFSMSdfp: Using DFSMSdfp in the z/OS Environment, SC26-7473
• z/OS DFSMSdss Diagnosis Reference, KY27-7618
• z/OS DFSMS: Implementing System-Managed Storage, SC27-7407
• z/OS DFSMS: Macro Instructions for Data Sets, SC26-7408
• z/OS DFSMS: Managing Catalogs, SC26-7409
• z/OS MVS: Program Management User’s Guide and Reference, SA22-7643
• z/OS MVS Program Management: Advanced Facilities, SA22-7644
• z/OS DFSMSdfp Storage Administration Reference, SC26-7402
• z/OS DFSMS: Using Data Sets, SC26-7410
• DFSMS/MVS: Using Advanced Services, SC26-7400
• DFSMS/MVS: Utilities, SC26-7414

DFSORT™
• DFSORT Application Programming: Guide, SC33-4035
• DFSORT Installation and Customization, SC33-4034

Distributed Relational Database Architecture
• Open Group Technical Standard; the Open Group presently makes the following DRDA books available through its Web site at www.opengroup.org
  – Open Group Technical Standard, DRDA Version 3 Vol. 1: Distributed Relational Database Architecture
  – Open Group Technical Standard, DRDA Version 3 Vol. 3: Distributed Data Management Architecture

Domain Name System

Education
• Information about IBM educational offerings is available on the Web at http://www.ibm.com/software/sw-training/
• A collection of glossaries of IBM terms is available on the IBM Terminology Web site at www.ibm.com/ibm/terminology/index.html

eServer zSeries
• IBM eServer zSeries Processor Resource/System Manager Planning Guide, SB10-7033

Fortran: VS Fortran
• VS Fortran Version 2: Language and Library Reference, SC26-4221
• VS Fortran Version 2: Programming Guide for CMS and MVS, SC26-4222

High Level Assembler
• High Level Assembler for MVS and VM and VSE Language Reference, SC26-4940
• High Level Assembler for MVS and VM and VSE Programmer’s Guide, SC26-4941

ICSF
• z/OS ICSF Overview, SA22-7519
• Integrated Cryptographic Service Facility Administrator’s Guide, SA22-7521

IMS Version 8

IMS product information is available on the IMS Library Web page, which you can find at www.ibm.com/ims
• IMS Administration Guide: System, SC27-1284
• IMS Administration Guide: Transaction Manager, SC27-1285

• IMS Application Programming: Database Manager, SC27-1286
• IMS Application Programming: Design Guide, SC27-1287
• IMS Application Programming: Transaction Manager, SC27-1289
• IMS Command Reference, SC27-1291
• IMS Customization Guide, SC27-1294
• IMS Install Volume 1: Installation Verification, GC27-1297
• IMS Install Volume 2: System Definition and Tailoring, GC27-1298
• IMS Messages and Codes Volumes 1 and 2, GC27-1301 and GC27-1302
• IMS Open Transaction Manager Access Guide and Reference, SC18-7829
• IMS Utilities Reference: System, SC27-1309

General information about IMS Batch Terminal Simulator for z/OS is available on the Web at www.ibm.com/software/data/db2imstools/library.html

IMS DataPropagator
• IMS DataPropagator for z/OS Administrator’s Guide for Log, SC27-1216
• IMS DataPropagator: An Introduction, GC27-1211
• IMS DataPropagator for z/OS Reference, SC27-1210

ISPF
• z/OS ISPF Dialog Developer’s Guide, SC23-4821
• z/OS ISPF Messages and Codes, SC34-4815
• z/OS ISPF Planning and Customizing, GC34-4814
• z/OS ISPF User’s Guide Volumes 1 and 2, SC34-4822 and SC34-4823

Language Environment
• Debug Tool User’s Guide and Reference, SC18-7171
• Debug Tool for z/OS and OS/390 Reference and Messages, SC18-7172
• z/OS Language Environment Concepts Guide, SA22-7567
• z/OS Language Environment Customization, SA22-7564
• z/OS Language Environment Debugging Guide, GA22-7560
• z/OS Language Environment Programming Guide, SA22-7561
• z/OS Language Environment Programming Reference, SA22-7562

MQSeries®
• MQSeries Application Messaging Interface, SC34-5604
• MQSeries for OS/390 Concepts and Planning Guide, GC34-5650
• MQSeries for OS/390 System Setup Guide, SC34-5651

National Language Support
• National Language Design Guide Volume 1, SE09-8001

NetView®
• Tivoli NetView for z/OS Installation: Getting Started, SC31-8872
• Tivoli NetView for z/OS User’s Guide, GC31-8849

Microsoft ODBC

Information about Microsoft ODBC is available at http://msdn.microsoft.com/library/

Parallel Sysplex Library
• System/390 9672 Parallel Transaction Server, 9672 Parallel Enterprise Server, 9674 Coupling Facility System Overview For R1/R2/R3 Based Models, SB10-7033
• z/OS Parallel Sysplex Application Migration, SA22-7662
• z/OS Parallel Sysplex Overview: An Introduction to Data Sharing and Parallelism, SA22-7661
• z/OS Parallel Sysplex Test Report, SA22-7663

The Parallel Sysplex Configuration Assistant is available at www.ibm.com/s390/pso/psotool

PL/I: Enterprise PL/I for z/OS
• IBM Enterprise PL/I for z/OS Language Reference, SC27-1460
• IBM Enterprise PL/I for z/OS Programming Guide, SC27-1457

PL/I: PL/I for MVS & VM
• PL/I for MVS & VM Programming Guide, SC26-3113

SMP/E
• SMP/E for z/OS and OS/390 Reference, SA22-7772
• SMP/E for z/OS and OS/390 User’s Guide, SA22-7773

Storage Management
• z/OS DFSMS: Implementing System-Managed Storage, SC26-7407
• MVS/ESA Storage Management Library: Managing Data, SC26-7397

• MVS/ESA Storage Management Library: Managing Storage Groups, SC35-0421
• MVS Storage Management Library: Storage Management Subsystem Migration Planning Guide, GC26-7398

System Network Architecture (SNA)
• SNA Formats, GA27-3136
• SNA LU 6.2 Peer Protocols Reference, SC31-6808
• SNA Transaction Programmer’s Reference Manual for LU Type 6.2, GC30-3084
• SNA/Management Services Alert Implementation Guide, GC31-6809

TCP/IP
• IBM TCP/IP for MVS: Customization & Administration Guide, SC31-7134
• IBM TCP/IP for MVS: Diagnosis Guide, LY43-0105
• IBM TCP/IP for MVS: Messages and Codes, SC31-7132
• IBM TCP/IP for MVS: Planning and Migration Guide, SC31-7189

TotalStorage™ Enterprise Storage Server
• RAMAC Virtual Array: Implementing Peer-to-Peer Remote Copy, SG24-5680
• Enterprise Storage Server Introduction and Planning, GC26-7444
• IBM RAMAC Virtual Array, SG24-6424

Unicode
• z/OS Support for Unicode: Using Conversion Services, SA22-7649

Information about Unicode, the Unicode consortium, the Unicode standard, and standards conformance requirements is available at www.unicode.org

VTAM
• Planning for NetView, NCP, and VTAM, SC31-8063
• VTAM for MVS/ESA Diagnosis, LY43-0078
• VTAM for MVS/ESA Messages and Codes, GC31-8369
• VTAM for MVS/ESA Network Implementation Guide, SC31-8370
• VTAM for MVS/ESA Operation, SC31-8372
• z/OS Communications Server SNA Programming, SC31-8829
• z/OS Communications Server SNA Programmer’s LU 6.2 Reference, SC31-8810
• VTAM for MVS/ESA Resource Definition Reference, SC31-8377
WebSphere family

- WebSphere MQ Integrator Broker: Administration Guide, SC34-6171
- WebSphere MQ Integrator Broker for z/OS: Customization and Administration Guide, SC34-6175
- WebSphere MQ Integrator Broker: Introduction and Planning, GC34-5999
- WebSphere MQ Integrator Broker: Using the Control Center, SC34-6168

z/Architecture™

- z/Architecture Principles of Operation, SA22-7832

z/OS

- z/OS C/C++ Programming Guide, SC09-4765
- z/OS C/C++ Run-Time Library Reference, SA22-7821
- z/OS C/C++ User’s Guide, SC09-4767
- z/OS Communications Server: IP Configuration Guide, SC31-8875
- z/OS DCE Administration Guide, SC24-5904
- z/OS DCE Introduction, GC24-5911
- z/OS DCE Messages and Codes, SC24-5912
- z/OS Information Roadmap, SA22-7500
- z/OS Introduction and Release Guide, GA22-7502
- z/OS JES2 Initialization and Tuning Guide, SA22-7532
- z/OS JES3 Initialization and Tuning Guide, SA22-7549
- z/OS Language Environment Concepts Guide, SA22-7567
- z/OS Language Environment Customization, SA22-7564
- z/OS Language Environment Debugging Guide, GA22-7560
- z/OS Language Environment Programming Guide, SA22-7561
- z/OS Language Environment Programming Reference, SA22-7562
- z/OS Managed System Infrastructure for Setup User’s Guide, SC33-7985
- z/OS MVS Diagnosis: Procedures, GA22-7587
- z/OS MVS Diagnosis: Reference, GA22-7588
- z/OS MVS Diagnosis: Tools and Service Aids, GA22-7589
- z/OS MVS Initialization and Tuning Guide, SA22-7591
- z/OS MVS Initialization and Tuning Reference, SA22-7592
- z/OS MVS Installation Exits, SA22-7593
- z/OS MVS JCL Reference, SA22-7597
- z/OS MVS JCL User’s Guide, SA22-7598
- z/OS MVS Planning: Global Resource Serialization, SA22-7600
- z/OS MVS Planning: Operations, SA22-7601
- z/OS MVS Planning: Workload Management, SA22-7602
- z/OS MVS Programming: Assembler Services Guide, SA22-7605
- z/OS MVS Programming: Assembler Services Reference, Volumes 1 and 2, SA22-7606 and SA22-7607
- z/OS MVS Programming: Authorized Assembler Services Guide, SA22-7608
- z/OS MVS Programming: Authorized Assembler Services Reference Volumes 1-4, SA22-7609, SA22-7610, SA22-7611, and SA22-7612
- z/OS MVS Programming: Callable Services for High-Level Languages, SA22-7613
- z/OS MVS Programming: Extended Addressability Guide, SA22-7614
- z/OS MVS Programming: Sysplex Services Guide, SA22-7617
- z/OS MVS Programming: Sysplex Services Reference, SA22-7618
- z/OS MVS Programming: Workload Management Services, SA22-7619
- z/OS MVS Recovery and Reconfiguration Guide, SA22-762A
- z/OS MVS Routing and Descriptor Codes, SA22-7623
- z/OS MVS Setting Up a Sysplex, SA22-7625
- z/OS MVS System Codes SA22-7626
- z/OS MVS System Commands, SA22-7627
- z/OS MVS System Messages Volumes 1-10, SA22-7631, SA22-7632, SA22-7633, SA22-7634, SA22-7635, SA22-7636, SA22-7637, SA22-7638, SA22-7639, and SA22-7640
- z/OS MVS Using the Subsystem Interface, SA22-7642
- z/OS Planning for Multilevel Security and the Common Criteria, SA22-7509
- z/OS RMF User’s Guide, SC33-7990
- z/OS Security Server Network Authentication Server Administration, SC24-5926
- z/OS Security Server RACF Auditor’s Guide, SA22-7684
- z/OS Security Server RACF Command Language Reference, SA22-7687
- z/OS Security Server RACF Macros and Interfaces, SA22-7682
- z/OS Security Server RACROUTE Macro Reference, SA22-7692
- z/OS Support for Unicode: Using Conversion Services, SA22-7649
- z/OS TSO/E CLISTs, SA22-7781
- z/OS TSO/E Command Reference, SA22-7782
• z/OS TSO/E Customization, SA22-7783
• z/OS TSO/E Messages, SA22-7786
• z/OS TSO/E Programming Guide, SA22-7788
• z/OS TSO/E Programming Services, SA22-7789
• z/OS TSO/E REXX Reference, SA22-7790
• z/OS TSO/E User’s Guide, SA22-7794
• z/OS UNIX System Services Command Reference, SA22-7802
• z/OS UNIX System Services Messages and Codes, SA22-7807
• z/OS UNIX System Services Planning, GA22-7800
• z/OS UNIX System Services Programming: Assembler Callable Services Reference, SA22-7803
• z/OS UNIX System Services User’s Guide, SA22-7801
Index

A
access methods
  SNA
    descriptions 127
  TCP/IP
    descriptions 109
access methods, SNA
  overview 109
access methods, TCP/IP
  overview 108
access path, effects 272
accounting trace
  global lock contention 232
active log data set
  recommendation 30
affinity
  establishing for installation jobs 77
  for online utility jobs 149
affinity, breaking between systems 141
ALIAS option
  of DSNJU003 utility 119, 142
ALTER GROUPBUFFERPOOL command
  change group buffer pools 270
  GBPCACHE option 272
application design recommendations 202
AUTH AT HOP SITE parameter 71
authorization
  commands 146
  controlling access in a group 16
automatic
  rebinding, considerations in a coexistence environment 94
  rebuild of coupling facility structures
    description 168
    REBUILDPERCENT parameter of CFRM policy 40
  restart function of z/OS 36
availability
  catalog and directory 44
  coupling facility 37
  planning 35
  provided by data sharing 1
  volatility of coupling facility 41

B
BACKOUT DURATION option of panel DSNTPN 193
backout processing, postponing 193
backup database, recommendations 162
BACKUP SYSTEM utility 17, 164
batch processing, designing 202
binding, recommendations for a coexistence environment 95
BSDS (bootstrap data set)
  increasing size 63
  naming recommendation 30, 70
buffer pool
  allocating for parallel processing 216
  relationship to group buffer pool 9
  setting thresholds for Sysplex query parallelism 208
  tuning for Sysplex query parallelism 214

C
CACHE option of CREATE SEQUENCE statement 64
cache sequencing 64
cache structure
  name 32
CAF (call attachment facility)
  coexistence considerations 94
castout
  causes 251
  class queue 256
  description 251
  displaying castout owner 252
catalog alias 27
CATALOG ALIAS parameter 71
CFRM policy
  description 23
  REBUILDPERCENT parameter 40
CFSizer 48
change log inventory (DSNJU003) utility
  See DSNJU003 (change log inventory) utility
change log inventory utility
  specifying resynchronization port number 119
checkpoint, group buffer pool 253
CICS
  applications, migration considerations in a Sysplex 64
  attachment facility 60
  CICSPlex SM 64
coeexistence
  considerations for specific functions 98
  requirements for data sharing 92
cold start 195
COMMAND PREFIX parameter 70
commands
  authorizing 146
  prefix
    allowable characters 29
    displaying 151
    message format 61
    scope 145
communications database (CDB)
  identifying remote data sharing groups, SNA 130
  identifying remote data sharing groups, TCP/IP 116
  mapping DB2 location names 109
  updating 117, 131
conditional restart 194
configuration
  disk connectivity 26
  Parallel Sysplex components 21
  planning 21
  possibilities with data sharing 4
  z/OS system 32
collection request
  DB2 location name 108
connections
  displaying, group buffer pool 260
  exit routine, considerations 16
  failed-persistent 147
  maximum number of distributed 107
  monitoring connections to remote systems 157
  connectivity, checking 26
contention, false
avoiding 223
detecting 223
LOCK ENTRY SIZE parameter of installation panel
DSNTPJ 54
coupling facility
availability 37
channel failure 42
DiscFailCon event 191
failure-isolated 37
failures 38
non-failure-isolated 37
placement 38
planning for shutdown 197
recovery scenarios 173
reducing page size overhead 238
relation to INITSIZE 48
resource management policy.
See CFRM policy
RMF reports 200
storage
group buffer pool 48
lock 54
SCA 55
structures
allocation size 47
changing size of 47, 56
connection disposition 147
dealing with hung 191
displaying size and usage 151
estimating storage 47
monitoring 200
names 31
policy definition 23
structure disposition 147
types 22
volatile 41
Coupling Facility Activity Report of RMF 230, 259
Coupling Facility Structure Sizer (CFSizer) 48
CREATE SEQUENCE statement 64
CREATE TABLESPACE statement
MEMBER CLUSTER option 245
TRACKMOD option 245
critical data 44
cross-system coupling facility (XCF)
See XCF (cross-system coupling facility)
cross-system extended services (XES) 191
CURRENT MEMBER, special register 156
data sharing (continued)
requirements 19
data sharing group
access methods 107
accessing remote groups in SNA networks 137
accessing remote groups in TCP/IP networks 120
configuring as SNA server 132
for group-generic access 132
specifying generic LU name 133
configuring as TCP/IP server 117
designating subsets of members 119
registering names in DNS 119
specifying DRDA port number 118
specifying generic LU name 120
specifying resynchronization port number 119
creating 75
definition 1
description 1
identifying generic LU name 135
maintaining 18
member 1
shutting down network connection to requester 141
data sharing groups
identifying remote groups 116, 130
database
backup, recommendations 162
recovery 160
DATABASE PROTOCOL parameter 72
DB2 commands
routing 145
scope 145
DB2 Connect requester
configuring to use group access, TCP/IP 120
configuring to use group-generic access, SNA 139
requirement to enable Sysplex support 120, 139
DB2 GENERIC LUNAME parameter 131, 133, 134
DB2 location name
mapped to network element 108
DB2 LOCATION NAME parameter 72
DB2 location names
mapping to network element 109
DB2 NETWORK LUNAME parameter 71
DB2 Performance Expert
accounting report 232
data sharing reports 201
statistics report 267
DB2 private protocol
not recommended 107
DB2 release level
displaying 151
DB2 requester
configuring to use group access, TCP/IP 120
configuring to use group-generic access, SNA 140
configuring to use member-specific access, SNA 137
configuring to use member-specific access, TCP/IP 123
DBADM CREATE AUTH parameter 71
deadlock 226
DEALLOC PERIOD parameter 73
DECPSID parameter of DSNHDECP 75
DEFAULT BUFFER POOL FOR USER DATA parameter 73
DEFAULT BUFFER POOL FOR USER INDEXES parameter 73
deferred restart 195
deferring restart processing 195
DELETE NAME 261
DEVICE TYPE 1 parameter 73
DEVICE TYPE 2 parameter 73
DSMSHsm (Data Facility Hierarchical Storage Manager)
archiving logs in data sharing 159
directory entries of group buffer pool
description 263
symptoms of too few 266
disaster recovery, in a group 164
DiscFailConn (Disconnected/Failed Connection) event 191
disk, shared requirement 26
DISPLAY BUFFERPOOL command
LIST option 240
DISPLAY DATABASE command 226
DISPLAY GROUP command
displaying information about the group 150
group and member release level 92
DISPLAY GROUPBUFFERPOOL command
group detail report example 264
MDETAIL option 247
monitor group buffer pools 260
summary report example 260
DISPLAY THREAD command 211
displaying
information about
  coupling facility structures 151
data sharing group 150
LPL (logical page list) 154
retained locks 155
distributed data
connection limit in a data sharing group 107
moving to data sharing 83
recommendations for SNA alternatives 127
distributed data facility (DDF) 107
distribution libraries 76
DNS network addressing 113
element configuration 114
element of group access 122
example of member-specific access 125
domain name server (DNS) 113
domain name server (DNS), registering names 119
DRDA 107

DRDA port
updating using DSNJU003 utility 142
DRDA port number
recommendation 118
reserving 118
specifying 118

DRDA PORT parameter 72

DSNICOPY utility
merging DB2 subsystems 84
DSN3@ATH connection exit routine
See connection exit routine
DSN3@SGN sign-on exit routine
See sign-on exit routine
DSNDWQ04 mapping macro 245

DSNHDECP load module
  group attachment name 148
  shared 75
DSNJU003 (change log inventory) utility 142
  ALIAS option 119
designating subsets of members 119
specifying DRDA port number 118
DSNJU003 utility
  specifying resynchronization port number 119
DSNTEJ1 job 89
DSNTESD data set member 89
DSNTJFFT job 104
DSNTJGJF job 104
DSNTJJC job, data sharing member install 81
DSNTIP5 installation panel
  specifying DRDA port number 118
DSNZPxxx
installing 69
options 70
DUPEX
  option of CFRM policy 196
  parameter of CFRM policy 26
duplexed group buffer pools
See group buffer pool, duplexing
duplexing, benefits 35
dynamic VIPA network addressing 110
configuration requirements 112
dependencies 111
example configuration 111
example of group access 121
example of member-specific access 123
preparing for failure recovery 113

E
EDM pool
cross-invalidation 58
estimating storage 58
EDMPOOL STORAGE SIZE parameter 73
element names for ARM policy 36
estimating storage 46
exit routine, considerations 16
EXPLAIN statement, executing 273
explicit hierarchical locking 220
EXTENDED SECURITY parameter 72

F
failed-persistent connection
description 147
protects retained locks 190
failure scenario
  connectivity failure
group buffer pool 170
  lock structure and SCA 169
duplexed group buffer pool 172
  structure failure 171
false-isolated coupling facility 37
  fallback, in a group 100
false lock contention, preventing 54, 223
field procedure, considerations 16
  function level of IRLM 19, 93

G
GBP-dependent
See group buffer pool, dependency
GBPCACHE ALL clause
caching pages during a read 247
planning consideration 248
when to use 248
GBPCACHE attribute of group buffer pools
duplexed group buffer pools 272
procedure for altering 272
GBPCACHE CHANGED clause
defining table spaces and indexes 206
GBPCACHE clause
effect on guidelines for group buffer pool castout
  thresholds 258
GBPCACHE SYSTEM clause
  use for LOB table spaces 248
group buffer pool (continued)

- relationship to virtual buffer pools 9
- storage requirement 48
- storage shortage recovery scenario 178
- structure failure scenarios 172
- thresholds 255
- too few data entries 266
- too few directory entries 266
- too small 263
- tuning size and ratio 263
- write operations 248

GROUP ATTACH parameter 72

- group attachment name 72
- choosing 28
- DECPSSID parameter of DSNHDECP load module 75
- defining on IEFSSNx parmlib member 61
- DISPLAY command 92
- migration considerations 61
- submitting applications using 148
- overview 108

GROUPBUFFERPOOL 260

- assigning to a page set 237
- castout threshold guidelines 257
- changing size 270
- changing using ALTER GROUPBUFFERPOOL 270
- checkpoint 253
- connectivity failure scenarios 170
- cross-validation 236
- data entries 263
- default castout threshold 257
- dependency 239, 241
- description 22, 236
- determining inner-system sharing 241
- directory entries 263
- duplexing
  - altering ratio 272
  - connections 153
  - description 42
  - GBPCACHE attribute 272
  - performance 43
  - summary of failure scenarios 172
- failures 38
- monitoring using DISPLAY GROUPBUFFERPOOL 260
- monitoring using RMF report 259
- monitoring using z/OS command D XCFSTR 258
- name of cache structure 32
- naming conventions 32
- read operations 246
- rebuilding 40
- RECOVER-pending (GRECP) status
  - removing using START DATABASE command 176, 177

Hung coupling facility connections 191

ICF (Integrated Coupling Facility) 166

IEFSSNx parmlib member
- command prefix 61

IFCID (instrumentation facility component identifier)
- identifiers by number
  - 0045 233
  - 0221 213
  - 0222 213
  - 0231 213
  - 0250 178
  - 0251 245
  - 0259 245
  - 0267 41
  - 0268 41
- IMMEDIATE WRITE parameter 74

IMS, attachment facility 60

in-memory statistics, maintaining 273

INITSIZE parameter of CFRM policy
- example 25
- relation to SIZE 48

INSTALL DD CONTROL SUPT. parameter 72

installation
- choosing options to enable data sharing 75
- jobs, system affinity 77
locks (continued)
physical (continued)
   instrumentation data 244
   page 244
   retained state 243
   when obtained 239
   retained
   P-locks 243
   releasing 156
structure
   changing size 234
   displaying size and usage 151
   monitoring usage 229
   symptoms of storage shortage 180
   XCF message buffer size effect on resolving global contention 225
log
   active 158
   archive 158
   archiving using DFSMSshima 159
   names 30
log record
   applying 162
   header 162
log record sequence number (LRSN) 162
logical page list (LPL)
   description 154
   failed group buffer pool write 250
   recovering pages
   methods 168
   status, description 154
LU name of requestor
   identifying 136
LUWID option of DISPLAY THREAD command 157

M
maintenance, applying to a group 18
mapping macro, DSNDWQ04 245
MAX REMOTE ACTIVE option 142
MAX REMOTE CONNECTED thread limit 118
MAXDBAT parameter 142
MAXROWS option of CREATE TABLESPACE statement 244
MAXUSRS
   decreasing parameter of IRLMPROC 224
   effect on lock entry size 223
member
   configuring to use generic LU name 134
   definition 1
   preventing from processing DDF requests 142
   subsets 119
   unregistering from Workload Manager 142
MEMBER CLUSTER option of CREATE TABLESPACE 202, 245
member domain name, choosing for a group 29
MEMBER IDENTIFIER parameter 71
member name
   DB2 28
   displaying 151
   IRLM 28
MEMBER NAME parameter 71
member on which SQL statements run, determining 156
MEMBER parameter of ssnmMSTR startup procedure 78
member release level, displaying 151
member-scope, DB2 Performance Expert reports 201
member-specific access, SNA
   configuring DB2 requester 137
member-specific access, SNA (continued)
description 127
description example of 137
overview 109
RACF PassTicket limitation 128
switching from group-generic access 141
two-phase commit resynchronization 128
member-specific access, TCP/IP
configuring DB2 requester 123
description 115
example using DNS requester addressing 125
equivalent using dynamic VIPA network addressing 123
overview 108
merging DB2 subsystems 83
message by identifier
DSN750A 175
DSN7502 177
DSN7503 177
DSN7505I 179
DSN7512A 179
DSNB228I 173, 175, 178
DSNB250E 173, 175
DSNB301E 175, 178
DSNB303E 173, 175
DSNB304I 174
DSNB305I 174
DSNB309I 175
DSNB311I 175
DSNB312I 175
DSNB313I 175
DSNB314I 173, 176, 177
DSNB319A 178
DSNB320I 174
DSNB321I 174
DSNB325A 178
DSNB353I 174
DSNB354I 174
DSNJO06I 168
DSNJOI02I 168
DSNJO22I 168
DSNNJ46I 104
DXR136I 175, 177
DXR142E 180
DXR170I 180
migration of a group 91
MINIMUM DIVIDE SCALE parameter 72
N
naming
coupling facility structures 31
description example of 34
group names 27, 275
IRLM 30
IRLM names 276
member names 27, 275
recommendations 32
network protocol
choosing 131
network protocol, choosing 109, 117
NOCACHE option of CREATE SEQUENCE statement 64
non-failure-isolated coupling facility 37
notices, legal 279
NSYSLLX parameter of IEASYSxx parmlib member 26
P
P-locks, page
reducing contention 244
space map contention 245
page
locks, physical 244
validity test in data sharing 246
page set
determining if group buffer pool dependent 241
P-lock, determining retained state 243
page size
reducing coupling facility overhead 238
parallel queries
monitoring 211
tuning 211
Parallel Sysplex
definition 1
Parallel Sysplex requirements 21
parallel tasks
displaying information about 211
using a performance monitor 212
parallelism assistant
definition 203
parallelism coordinator
definition 203
parallelism, Sysplex query 203
PARAMETER MODULE parameter 71
partitioned table space, locking 225
performance
designing table spaces 202
improving 201
migrating batch applications 202
recommendation 202
setting expectations 199
performance trace
global lock contention 233
monitor structure rebuild 41
phases of restart 186
physical lock
description 238
instrumentation data 244
retained state 243
when obtained 239
PLAN_TABLE table, GROUP_MEMBER column 273
planning XA
Global Transactions 45
xa 45
policy
e automatic restart (ARM) 36
CFRM (coupling facility resource management) 23
SFM (Sysplex failure management) 40
PORT option
of DSNJU003 utility 142
postponing backout 193
prefetch processing
for GBP-dependent page sets and partitions 247
PREFLIST parameter of CFRM policy 25
priority
of parallel query work 216
PROCNAME parameter 71
Q
query applications, designing for data sharing 203
QXREPOL field of statistics and accounting traces 212
QXREPOP2 field of statistics and accounting traces 213
R

RACF PassTickets
   generic LU name 137
   generic LU name requirement 120
   location alias limitation 128, 137
   ratio of directory entries to data entries
   changing 272
   choosing a value 263
   description 263
   re-enabling DB2 data sharing 105
   READ COPY2 ARCHIVE parameter 74
   READ TAPE UNITS parameter 74
   reason codes
      X'00C20204' 175, 178
      X'00C20205' 176, 177
      X'00C20220' 175
      X'00F70609' 179
   rebinding automatically
      access path change in data sharing 273
   rebinding, automatically
      avoiding in a coexistence environment 94
   rebuild structures
      monitoring time for structure rebuild 41
      specifying rebuild threshold 40
      using z/OS command SETXCF START,REBUILD 56
   REBUILDPERCENT parameter of CFRM policy 40
   RECORDING MAX parameter 74
   RECOVER TABLESPACE utility, options
      TOCOPY 163
      TOLOGPOINT 163
      TORBA 163
   RECOVER utility 17
   recovering databases 160
   recovery
      after disaster 164
      automatic recovery of group buffer pools 39
      coupling facility 168
      description 161
      LPL (logical page list) 168
      options 163
      point-in-time 163
      preparing for fast 162
      to currency 163
   registration tables for DDL, recommendation 72
   remote data sharing groups
      identifying 116, 130
   reports
      Coupling Facility reports of RMF 200
      group detail report 264
      member-scope reports of DB2 Performance Expert 201
      Response Time Distribution report of RMF 200
      Shared Device report of RMF 200
      summary report 260
      Sysplex Summary report of RMF 200
   requester
      definition 107
      setting up DB2 UDB for z/OS 115, 129
      shutting down network connection to data sharing group 141
   requester LU name
      identifying 136
   requirements for data sharing 19
   RESET GENERICLU command 141
   resource limit facility (governor)
      data sharing 203
      Sysplex query parallelism 217
   resource limit specification table (RLST)
      command scope 203
      unique name 203
   resource measurement facility (RMF)
      Coupling Facility Activity Report 259
      reports for data sharing 200
   RESREPORT option
      of DSNJU003 utility 143
   restart
      automatic 36
      conditional 194
      DB2 183
      deferring processing 195
      group 183
      light 36, 185
      postponing backout processing 193
   RESTORE SYSTEM utility 18, 164
   RESYNC PORT parameter 71
   resynchronization port
      updating using DSNJU003 utility 143
   resynchronization port number
      specifying 119
   resynchronization port, choosing for a group 29
   RETAINED LOCK TIMEOUT parameter 75
   retained utility ID lock 184
   reverting to simplex mode of structure 196
   RMF (resource measurement facility)
      Coupling Facility Structure Activity report 230
      monitor group buffer pools 259
      reports for data sharing 200
      row locking, page P-lock contention 244

S

SAF (security authorization facility) class for coupling facility structures 26
sample application for data sharing 89
SCA (shared communications area)
   description 22
   displaying size and usage 151
   increasing storage 179
   name of list structure 31
   storage requirement 55
   scope of commands 145
   SCOPE parameter of IRLMPROC, NODISCON option 19
   SDSNSAMP library, naming recommendation 82
   selective partition locking
      description 225
      monitoring 226
   SETSSI command of z/OS 82
   SETXCF START,ALTER command of z/OS 56
   SETXCF STOP, REBUILD command of z/OS 196
   SFM (Sysplex failure management) policy of z/OS 40
   shared data, restricting access to 238
   shared disk, requirement 26
   SHAREPORT option, TCP/IP PORT configuration profile
      statement 118
   sign-on exit routine, considerations 16
   simplex mode of structure, reverting 196
   single-member access, SNA
      description 129
      overview 109
   single-member access, TCP/IP
      description 115
      overview 108
   SITE TYPE parameter 72
   SIZE parameter of CFRM policy 25
SYS1.PROCLIB
SYS1.PARMLIB
SUBSYSTEM
structure
storage
STOP
status
statistics
START
START
SPUFI,
space
SNA
SNA
SNA
command
avoiding
restrictions
configuring
global
restrictions
parameters
description
avoiding
IRLM
LPL
displaying
definition
127
SN server
configuring data sharing group as
for group-generic access
specifying generic LU name
space map, reducing P-lock contention
SPUFI, recommendations for a coexistence environment
START DATABASE
restrictions
START IRLM TRACE parameter
startup, DB2
STATISTICS HISTORY parameter
statistics trace
global locking
status
displaying member status
GRECP
LPL
STOP DATABASE
restrictions
STOP DDF command
storage
coupling facility
calculating
duplexing
group buffer pool
lock structure
shared communications area (SCA)
EDM pool
changes for data sharing
IRLM
storm drain effect
avoiding with a CICS enhancement
avoiding with a CICS exit
description of
structure
duplexing
starting
stopping
subsets of members
SUBSYSTEM
name
for data sharing group members
for DB2 group members
for IRLM in data sharing
parameters
load module
recommendations
SUBSYSTEM NAME (IRLM) parameter
SUBSYSTEM NAME parameter
SUBSYSTEM SEQUENCE parameter
summary report, DISPLAY GROUPBUFFERPOOL
command
SYSLIB library
SYSTPROCLIB library
SYSTI.PARMLIB
SYSTI.PROCLIB
SYSBMI.IPLIST
for member-specific access, TCP/IP
for single-member access, TCP/IP
for TCP/IP access
SYSBMI.IPNAMES
for group access, TCP/IP
for member-specific access, TCP/IP
for single-member access, TCP/IP
for TCP/IP access
SYSBMI.LOCATIONS
for group access, TCP/IP
for member-specific access, SNA
for member-specific access, TCP/IP
for single-member access, TCP/IP
for SNA access
for TCP/IP access
SYSBMI.LULIST
for member-specific access, SNA
for single-member access, SNA
for SNA access
SYSBMI.LUNAMES
for group-generic access, SNA
for member-specific access, SNA
for single-member access, SNA
for SNA access
GENERIC column
SYSLGRNX table space
increasing its size
Sysplex
definition
Sysplex Distributor
role in dynamic VIPA network addressing
Sysplex domain name, choosing for a group
SYSPLEX parameter, DB2 Connect requester
Sysplex query parallelism
accounting report
configuration requirements
data set placement
description
designing a database for
determining parallel degree
disabling
enabling
improving response time
installation verification test
IRLM storage considerations
monitoring processor use
planning for
setting limits
setting workload management goals
example
Sysplex requirements for data sharing
Sysplex Timer
SYSTEM ADMIN 1 parameter
SYSTEM ADMIN 2 parameter
SYSTEM OPERATOR 1 parameter
SYSTEM OPERATOR 2 parameter
system-level point-in-time recovery

T

table spaces
LOB
use GBPCACHE SYSTEM clause
target library

332 Data Sharing: Planning and Administration
TCP/IP access methods 108, 109
   group access, description 110
   group access, overview 108
member-specific access, description 115
member-specific access, overview 108
recommendation 108
single-member access, description 115
single-member access, overview 108
connecting distributed partners 120
definition 109
DNS network addressing 113
example configuration 114
example of group access 122
example of member-specific access 125
dynamic VIPA network addressing 110
configuration requirements 112
example configuration 111
example of group access 121
example of member-specific access 123
preparing for failure recovery 113
TCP/IP ALREADY VERIFIED parameter 72
TCP/IP server
   configuring data sharing group as 117
   designating subsets of members 119
   registering names in DNS 119
   specifying DRDA port number 118
   specifying generic LU name 120
   specifying resynchronization port number 119
threads
   monitoring 157
threshold, group buffer pool
castout 255
class castout default 256
defaults 257
guidelines 257
timeout in a group 226
trace
   accounting 232
   events in a group 201
   statistics, global locking activity 231
TRACKER SITE parameter 72
TRACKMOD option of CREATE and ALTER TABLESPACE 202, 245
TSO connections, coexistence considerations 94
two-phase commit processing
   using group-generic access, SNA 128
two-phase commit resynchronization
   using member-specific access, SNA 128

U
utilities
   identifier 150
   recommendations for a coexistence environment 95
   stand-alone 150
   stopping and restarting 149
   submitting to a group 149
   work data sets 150
utility ID lock, retained 184

V
validation routine, considerations 16
VARCHAR FROM INDEX parameter 72
VARY NET;INACT;JD command 141
volatility of coupling facility 41
VPPSEQT
   definition 216
VPXPSEQT
   buffer pool threshold 208
   definition 216
VTAM (Virtual Telecommunications Access Method)
   breaking affinity between systems 141
generic resources 128

W
work file database
   connectivity requirements 26
   considerations when disabling data sharing 104
   considerations when re-enabling DB2 data sharing 105
   name 29
WORK FILE DB parameter 71
workload
   management facility of z/OS
   period switches on parallelism assistants 205
   setting goals for parallel queries 204
Workload Manager (WLM)
   role in DNS network addressing 114
   role in dynamic VIPA network addressing 110
   role in member-specific access 115
   unregistering members 142

X
XCF (cross-system coupling facility) component of z/OS
   DB2 group name 27
   description 21
   IRLM group name 30
   message buffer size effect on resolving global contention 225
   signal contention 216
XES (cross-system extended services) 191
XRMIOUT exit of CICS 65

Z
z/OS commands
   D XCF 152
   SETSSI 82
Readers’ Comments — We’d Like to Hear from You

DB2 Universal Database for z/OS
Data Sharing:
Planning and Administration
Version 8

Publication No. SC18-7417-03

Overall, how satisfied are you with the information in this book?

<table>
<thead>
<tr>
<th></th>
<th>Very Satisfied</th>
<th>Satisfied</th>
<th>Neutral</th>
<th>Dissatisfied</th>
<th>Very Dissatisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall satisfaction</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>

How satisfied are you that the information in this book is:

<table>
<thead>
<tr>
<th></th>
<th>Very Satisfied</th>
<th>Satisfied</th>
<th>Neutral</th>
<th>Dissatisfied</th>
<th>Very Dissatisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accurate</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>Complete</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>Easy to find</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>Easy to understand</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>Well organized</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>Applicable to your tasks</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you?  □ Yes  □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.