IBM DB2 Connect User's Guide Version 5

Document Number S10J-8163-00

IBM DB2 Connect

User's Guide

Version 5

S10J-8163-00

IBM DB2 Connect

User's Guide

Version 5

S10J-8163-00

Before using this information and the product it supports, be sure to read the general information under Appendix E, “Notices” on page 103.

Order publications through your IBM representative or the IBM branch office serving your locality or by calling 1-800-879-2755 in U.S. or 1-800-IBM-4YOU in Canada.

Contents

About This Book

This book contains general information about the following IBM DB2 Connect products, formerly known as Distributed Database Connection Services (DDCS):

- DB2 Connect Personal Edition (formerly known as DDCS Single-User).
- DB2 Connect Enterprise Edition, for AIX, HP-UX, Solaris, OS/2, and Windows NT (formerly known as DDCS Multi-user Gateway).

This book is intended for programmers and administrators who are responsible for setting up and maintaining DB2 Connect connections between DB2 Universal Database clients and any of the following DRDA application server database management systems:

- DATABASE 2 for OS/390 (DB2 for OS/390) Version 5
- DATABASE 2 for MVS (DB2 for MVS) Version 3 or higher
- DATABASE 2 for VSE and VM (DB2 for VSE & VM)
- DATABASE 2 for AS/400 (DB2 for AS/400)
- Any other relational database management system that implements Distributed Relational Database Architecture (DRDA) application server function.

Notes:

1. DB2 Universal Database Version 5.0 does not require DB2 Connect in order to function as a DRDA application server.
2. DB2 for OS/390 Version 5.1 or higher is required in order to use DRDA Level 3 functions, including TCP/IP database connections, and stored procedures with multi-row answer sets.

This book explains concepts that are applicable to both DB2 Connect products. For information about a specific platform (for example, OS/2, AIX, or HP-UX), refer to either:

- *DB2 Connect Personal Edition Quick Beginnings*
- *DB2 Connect Enterprise Edition Quick Beginnings*.

This book also describes features of IBM DB2 Connect products and provides general information about setting up connections to DRDA server databases. In order to fully understand the configuration and communication protocols, you should be familiar with Systems Network Architecture (SNA) protocols, TCP/IP, DRDA, and your DRDA server operating system.

DB2 Connect is part of the DB2 Universal Database family of products.

How This Book is Structured

This book contains the following major sections:

- Chapter 1, “DB2 Connect Overview” on page 1
- Chapter 2, “Distributed Relational Database Architecture Concepts” on page 9

Other Information Sources

This section lists other information sources which may be useful.

Using the World Wide Web

You can find the most recent information about DB2 Connect, DB2 Universal Database, and other IBM software products on the World Wide Web. This includes the latest publications, as well as technical hints and tips in the form of Technotes:

1. Set your Web browser to the following URL:

```
http://www.software.ibm.com/data/db2/library/
```

2. Select “DB2 Universal Database”.
3. For example, search for “Technotes” using the keywords “DDCS”, “DRDA”, or “Connect”.

Related DRDA Publications

The following books contain related information and may be referenced in this manual.

Form number	Book title
SC26-4783	*Distributed Relational Database Architecture Connectivity Guide*
SC26-4773	*Distributed Relational Database Architecture Application Programming Guide*
SC26-4782	*Distributed Relational Database Architecture Problem Determination Guide*
SC26-4650	*Planning for Distributed Relational Database Architecture*
GC26-3195	*Distributed Relational Database Architecture Every Manager's Guide*
G321-5482	*IBM Distributed Data Management Architecture Level 3: Reference*

Related DRDA Server Publications

Related DRDA server publications include the following books from the DB2 for AS/400, DB2 for MVS/ESA, and DB2 for VSE & VM libraries.

Form number	Book title
SC41-5702	*AS/400 Distributed Database Programming*
SC41-9609	*AS/400 SAA Structured Query Language/400 Programmer's Guide*
SC41-9608	*AS/400 SAA Structured Query Language/400 Reference*
GC21-8180	*AS/400 Communications Configuration Reference*
SC26-8958	*DB2 for OS/390 Application Programming and SQL Reference*
SC26-8960	*DB2 for OS/390 Command Reference*
GC26-8970	*DB2 for OS/390 Installation Reference*
SC26-8964	*DB2 for OS/390 Reference for Remote DRDA Requesters and Servers*
SC26-8966	*DB2 for OS/390 SQL Reference*
SC26-8957	*DB2 for OS/390 Administration Guide*
SC26-8967	*DB2 for OS/390 Utility Guide and Reference*
SH09-8087	*DB2 for VSE & VM SQL Reference*
SC26-3255	*IBM SQL Reference*

Other Related Publications

Form number	Book title
SG24-2212	*DRDA Support for TCP/IP in DB2 for OS/390 V5.1 and DB2 Universal Database V5.0*
SC33-0814	*CICS for AIX Application Programming Guide*
SC33-0931	*CICS for AIX Customization and Operation Guide*
S10J-7888	*DB2 Connect Enterprise Edition Quick Beginnings*
S10J-8162	*DB2 Connect Personal Edition Quick Beginnings*
GG24-4308	*DDCS for OS/2 to DB2 Performance Benchmark*
GG24-3771	*Distributed Relational Database Architecture: Using OS/2 DRDA Client Support with DB2*

Form number	Book title
GG24-3926	*OS/2 DDCS and DB2 V2.3 Distributed relational Database Performance: Early Experience*
GG24-4155	*Distributed Relational Database Architecture: Using DDCS for AIX DRDA support with DB2 for MVS/ESA and DB2 for AS/400*
GG24-4311	*Distributed Relational Database Architecture Cross Platform Connectivity and Application*
SC23-2443	*Encina for AIX Product Family Overview*
SC31-8094	*SNA Server for AIX and SNA Server Gateway for AIX Performance Guide*

Chapter 1. DB2 Connect Overview

Many organizations now use personal computers and UNIX workstations extensively, whilst at the same time still keeping much of their most important data on mainframes or minicomputers. Providing access to enterprise data for applications running on PC and UNIX applications is a significant challenge.

IBM DB2 Connect products provide an elegant solution to this situation. They enable database application programs running on DOS, Macintosh, OS/2, UNIX and Windows computers to access data stored in relational databases on MVS, OS/400, OS/390, VSE and VM hosts. Data stored in certain non-relational databases such as IMS can also be accessed when DB2 Connect is used in conjunction with data replication products such as IBM Data Propagator Non-Relational.

You can also use the DB2 Universal Database Client Application Enabler which is included in DB2 Connect to access databases stored on OS/2, Windows, and UNIX systems.

DB2 Connect provides applications with transparent online access to data by implementing a standard architecture for managing distributed data, Distributed Relational Database Architecture (DRDA). For example, you can use DB2 Connect with:

Spreadsheets, such as Lotus 1-2-3** and Microsoft Excel**, to analyze real-time data without the cost and complexity of data extract and import procedures
Decision support tools such as Intersolv Q+E Database Editor, and Crystal Reports, to provide real-time information
Database products such as MS Access** and Lotus Approach**
Development tools such as IBM VisualAge, PowerBuilder, Microsoft VisualBasic**, WATCOM VX-REXX, and VisPro/REXX, to create client/server solutions.
Internet servers such as Lotus Domino, Netscape Enterprise Server, and Microsoft Internet Information Server.
Java tools such as IBM Visual Java, and Symantec Visual Cafe Pro.
IBM Net.Data.

The workstation on which DB2 Connect is installed is called the *DB2 Connect workstation*. DRDA application servers can be accessed by local clients on the DB2 Connect workstation, and by remote clients. For remote clients, DB2 Connect acts as an intermediary between the client workstation and the DRDA server database management system.

Figure 1 on page 2 illustrates how DB2 Connect connects to other systems.

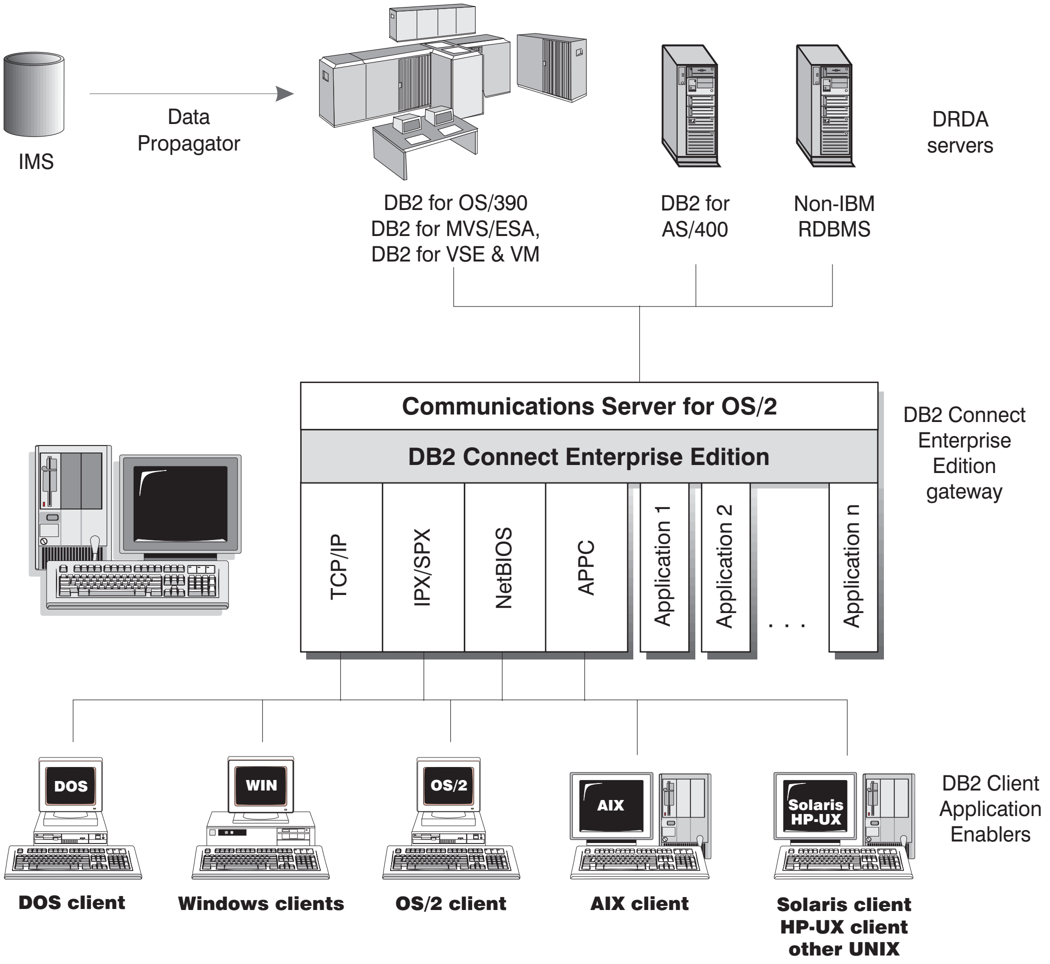

Figure 1. DB2 Connect Connections (OS/2 Example)

Notes:

1. This figure illustrates DB2 Connect Enterprise Edition running on an OS/2 system. DB2 Connect Enterprise Edition is implemented similarly on all supported platforms.
2. This figure includes remote clients. If you do not want remote clients, you do not need to install remote client support. In OS/2 and Windows environments, IBM also offers DB2 Connect Personal Edition, which supports a single user at a time, and does not provide gateway support for downstream remote clients.
3. This figure illustrates remote connections using TCP/IP, APPC, IPX/SPX, or NetBIOS. Depending on the operating system of the DB2 Connect workstation and the operating system of the remote client, different communication protocols may be available. For detailed information about communication protocol support, see *DB2 Connect Personal Edition Quick Beginnings*, or *DB2 Connect Enterprise Edition Quick Beginnings*.
4. You do not need to have DB2 Universal Database installed on the DB2 Connect workstation. If you want a complete relational database management system on the DB2 Connect workstation, order DB2 Universal Database.
5. DB2 Connect does not provide application development tools; however, IBM does provide DB2 software developer kits (SDKs) for DOS, Macintosh, OS/2, UNIX and Windows platforms. You can use these kits to develop applications that work with DB2 Connect. SDKs provide programming tools, utilities, documentation, and code samples. See *Building Applications for Windows and OS/2 Environments*, or *Building Applications for UNIX Environments* as appropriate for your platform.
6. C programmers developing Windows applications that use the Microsoft** ODBC interface should use the *Microsoft Open Database Connectivity Software Development Kit*. Programmers who want to develop applications using the COBOL programming language can use Micro Focus** COBOL and other development tools offered by Micro Focus, Inc. and IBM. In addition, there are a variety of 4GL development environments, application generators, CASE and other tools available from a number of vendors that can be used to develop applications that use DB2 Connect.

The Database Concept

The term *database* is used throughout this book to describe a relational database management system (RDBMS). Other systems with which DB2 Connect communicates may use the term *database* to describe a slightly different concept. The DB2 Connect term *database* is equivalent to:

MVS A DB2 for MVS/ESA subsystem identified by its LOCATION NAME

OS/390 A DB2 for OS/390 subsystem identified by its LOCATION NAME

VSE DB2 for VSE running in a partition identified by its DBNAME

VM DB2 for VM running in a CMS virtual machine identified by its DBNAME

OS/400 DB2 for AS/400, an integral part of the OS/400 operating system. Only one database can exist on an AS/400 machine. If the database will be used by applications outside the AS/400 system, the database must be given a name in the relational database directory.

Setting up DB2 Connect

Before you can use DB2 Connect, you must do the following:

1. Install DB2 Connect, and configure the DRDA server and workstation communications, as described in either *DB2 Connect Personal Edition Quick Beginnings*, or *DB2 Connect Enterprise Edition Quick Beginnings*.
2. Update the database directories as described in Chapter 3, “Updating Database Directories” on page 13. On OS/2, Windows 95, or Windows NT this can be done using the DB2 Universal Database Client Configuration Assistant, and on Windows 3.1 or Windows for Workgroups 3.11 this can be done using the Data Sources Setup dialog. On all other platforms the database directories must be updated using the DB2 Command Line Processor (CLP). Both approaches are described in *DB2 Connect Personal Edition Quick Beginnings*, and *DB2 Connect Enterprise Edition Quick Beginnings*.
3. Bind the DB2 Connect utilities to each DRDA server database management system, as described in Chapter 4, “Binding Applications and Utilities” on page 19. This task can also be performed using the DB2 Universal Database Client Configuration Assistant or the Data Sources Setup dialog where provided.

DB2 Connect and SQL

DB2 Connect forwards SQL statements submitted by application programs to DRDA servers. DB2 Connect can forward almost any valid SQL statement. The exceptions are documented in “DRDA Server SQL Statements Rejected by DB2 Connect” on page 57.

There are two types of SQL processing: static SQL and dynamic SQL. Static SQL minimizes the time required to execute an SQL statement by processing it ahead of time. Dynamic SQL is processed when the SQL statement is submitted to the DRDA Server. This makes dynamic SQL more flexible, but potentially slower. The decision to use static or dynamic SQL is made by the application programmer. Both are supported by DB2 Connect.

Different DRDA servers implement SQL differently. For information about common SQL statements that are supported by all IBM systems, see the *SQL Reference*.

DB2 Connect fully supports the common IBM SQL, as well as the DB2 for OS/390, DB2 for MVS/ESA, DB2 for VSE & VM (formerly SQL/DS), and DB2 for AS/400 implementations of SQL. IBM SQL is strongly recommended for maintaining database independence. For more information, see Chapter 7, “Programming in a DRDA Environment” on page 47.

Administration Utilities

The following utilities are available to help the DB2 Connect administrator:

- The command line processor lets you issue SQL statements against a DRDA server database. It flows the SQL statements to the database that you specify.
- The database system monitor utility lets the system administrator monitor system connections. It also helps the system administrator determine the source of an error. The system administrator can correlate client applications with the corresponding jobs running on the DRDA server.
- Import and export utilities let you copy data to and from a file on a workstation and a DRDA server database. These files can then be used for importing data into databases, spreadsheets, and other applications running on your workstation.
- The DB2 Connect trace utility lets application developers analyze the flow of the DRDA data stream between the DB2 Connect workstation and the DRDA server database management system.
- The DB2 Control Center lets administrators create and schedule data replication jobs as well as perform other database administration tasks.
- The DB2 Command Center provides a more graphical version of the command line processor.

For more information about these utilities, see Chapter 5, “DB2 Connect Administration Utilities” on page 25.

What's New in DB2 Connect Version 5?

- New easier to purchase packaging:
  - A single DB2 Connect Personal Edition package that contains OS/2, Windows 3.1, Windows 95, and Windows NT versions of the product. This package contains everything that is needed to get started, including a complimentary copy of Lotus Approach.
  - A single DB2 Connect Enterprise Edition package that contains OS/2, Windows NT, and all UNIX versions.
- Capability:
  - New Level 3 ODBC driver with many improvements
  - Updated JDBC driver for better Java support
  - Support for stored procedures that return multi-row result sets and multiple result sets (requires DB2 for OS/390 Version 5.1)
  - Built-in replication support
  - Generic bind option: you can specify any bind option supported by the host database.
  - SYSPLEX exploitation (DB2 Connect Personal Edition only; requires DB2 for OS/390 Version 5.1)

- Usability:
  - New installation method
  - TCP/IP database connections are much easier to configure (requires DB2 for OS/390 Version 5.1)
  - Integrated SNA support with point-and click configuration (DB2 Connect Personal Edition only)
  - New point and click configuration utility for configuring host connections.
  - Much easier process for connecting desktop client systems to DB2 Connect Enterprise Edition servers. Clients can discover DB2 Connect servers and all of the databases that are defined on each server
  - Improved ODBC traces with detailed information for performance analysis
  - Control Center and other GUI tools that simplify several DBA tasks
- Security:
  - DCE security (requires DB2 for OS/390 Version 5.1)
  - Ability to run ODBC applications without having to authorize each user to base tables. Users can now bind their ODBC driver in such a way as to allow applications to run under the authority of the person that bound the ODBC driver.
- Performance:
  - Faster access to the DB2 catalog for ODBC applications
  - Reduced network traffic:
    - Early close for cursors
    - Deferred prepare
    - Reduced byte count on Compound SQL
    - Several other network flow enhancements
    - Support for ASCII storage on the host (requires DB2 for OS/390 Version 5.1)
- Connectivity:
  - Support for DRDA over TCP/IP connections to other IBM DRDA Application Servers, as they introduce support for TCP/IP. (Native TCP/IP support currently requires DB2 for OS/390 Version 5.1)
  - SNA over TCP/IP via integrated MPTN support (requires AnyNet on the host).
  - Support for additional SNA connectivity options:
    - IBM Communication Server for Windows NT
    - IBM Personal Communications
- Other:

  - Ability to initiate 2-phase commit transactions over TCP/IP (requires DB2 for OS/390 Version 5.1)
  - Ability for desktop applications to participate in a 2-phase commit transactions without the need for a gateway (TCP/IP only, requires DB2 for OS/390 V5.1)
  - Ability to use DB2 for OS/390 for added reliability of transaction coordination (requires DB2 for OS/390 Version 5.1 and TCP/IP)
  - Numerous other enhancements and fixes affecting all aspects of system performance, reliability, and usability.

Functions Delivered in Previous Releases

DDCS Version 2 Release 4

Distributed Database Connection Services (DDCS) for Windows Single-User Version 2.4 introduced:

- A Data Sources Setup tool to help you define connections to host and AS/400 servers quickly and easily.
- Integrated SNA Support, to provide you with the communications support required to make these connections.
- A DB2 Password Expiration Maintenance utility (DB2PEM), which enables you to change your DB2 for MVS/ESA password without logging on to TSO.
- Enhancements to improve the performance and flexibility of DB2 Connect:
  - Deferred Prepare, which improves the performance of ODBC and other dynamic SQL applications by attaching the PREPARE request to a subsequent request instead of sending it separately.
  - Asynchronous ODBC, which improves the availability of ODBC applications. Previously, these might have appeared to be delayed while processing long queries in some network situations.
  - On AIX and OS/2, support for multi-threaded applications, which gives non-ODBC applications the ability to maintain multiple database connections with their own contexts.

DDCS Version 2 Release 3

New features in DDCS Version 2 Release 3.1 included:

- Two-phase commit for DRDA connections using the LU6.2 Syncpoint Manager (SPM) on OS/2 and AIX.

New features in DDCS Version 2 Release 3.0 included:

- Client application performance could be improved by running stored procedures on DB2 for MVS/ESA Version 4.1 and DB2 for AS/400 Version 3.1 servers. See “Stored Procedures” on page 54.

- Able to work with multiple databases in a single transaction. See “Distributed Unit of Work” on page 10.
- Able to improve performance by concatenating SQL statements. See “NOT ATOMIC Compound SQL” on page 55 and “Using Import and Export Utilities” on page 29.
- Able to implement chargeback accounting by using accounting strings. See “Implementing Chargeback Accounting” on page 58.
- Able to use many new bind options when binding applications to a DRDA application server. See “The BIND Command” on page 24.
- When using a DCE directory, the ability to consolidate the directory information needed by all your clients in a central repository. See Appendix C, “Using DCE Directory Services” on page 91.
- Greater flexibility in SQLCODE processing. See Chapter 8, “SQLCODE Mapping” on page 61.
- Diagnostic information stored in a readable format and consolidated in a single location (the first failure service log). For more information, see the *Troubleshooting Guide*.
- The DDCSSETP environment variable was replaced by BIND and PREPARE options such as `SQLERROR CONTINUE`, simplifying operations.
- Various other performance improvements were also implemented.

Chapter 2. Distributed Relational Database Architecture Concepts

Distributed Relational Database Architecture (DRDA) is a set of protocols that permits multiple database systems, both IBM and non-IBM, as well as application programs to work together. Any combination of relational database management products that use DRDA can be connected to form a distributed relational database management system. DRDA coordinates communication between systems by defining what must be exchanged and how it must be exchanged.

DRDA and DB2 Connect

DB2 Connect implements the DRDA architecture to reduce the cost and complexity of accessing data stored in DB2 for AS/400, DB2 for OS/390, DB2 for MVS/ESA, DB2 for VSE & VM, and other DRDA-compliant database servers. By fully exploiting the DRDA architecture, DB2 Connect offers a well-performing, low-cost solution with the system management characteristics that customers demand.

In DRDA terminology, an *application requester* is the code that handles the application end of a distributed connection; it is the application that is requesting data. An *application server* is the code that handles the database end of the connection. In the DB2 Connect environment, the DB2 Connect workstation functions as an application requester on behalf of application programs.

Figure 2 shows the flow of data between the DB2 Connect workstation and the DRDA server in the case where there are local clients only. In addition, a private protocol exists between the DB2 Connect workstation and any remote clients.

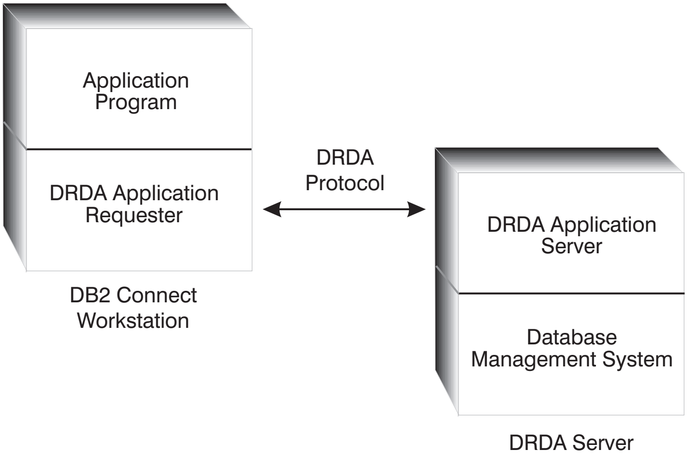

Figure 2. DRDA Flow in DB2 Connect

To implement the connections between DRDA server database management systems and database clients, DRDA uses the following architectures:

- Character Data Representation Architecture (CDRA)
- Distributed Data Management Architecture (DDM)
- Formatted Data Object Content Architecture (FD:OCA)
- Systems Network Architecture (SNA)
- SNA Management Services Architecture (MSA)
- Transmission Control Protocol/Internet Protocol (TCP/IP).

These are used as architectural building blocks. The data streams which flow over the network are specified by DRDA architecture, which documents a data stream protocol supporting distributed relational database access.

A request is routed to the correct destination by means of directories that contain various types of communication information and the name of the DRDA server database being accessed.

Remote Unit of Work

A unit of work is a single logical transaction. It consists of a sequence of SQL statements in which either all of the operations are successfully performed or the sequence as a whole is considered unsuccessful.

Remote unit of work lets a user or application program read or update data at one location per unit of work. It supports access to one database within a unit of work. While an application program can update several remote databases, it can only access one database within a unit of work.

Remote unit of work has the following characteristics:

- Multiple requests (SQL statements) per unit of work are supported.
- Multiple cursors per unit of work are supported.
- Each unit of work can access only one database.
- The application program either commits or rolls back the unit of work. In certain error circumstances, the database server or DB2 Connect may roll back the unit of work.

Distributed Unit of Work

Distributed unit of work allows an application to access more than one database within a unit of work; that is, the application can switch between databases before committing the data. This gives an application programmer the ability to do work involving multiple databases, local and remote, at the same time.

DB2 Connect always lets you read multiple databases within a unit of work. Whether you can update multiple databases depends on the products you are using:

- If you have SNA network connections, you can use the two-phase commit support provided with the Syncpoint Manager (SPM). The SPM is a part of DB2 Connect

Enterprise Edition Version 5.0 on OS/2 and AIX, and it enables you to to update multiple databases within a single unit of work for:

  - DB2 for MVS/ESA, Version 3.1 or later
  - DB2 for OS/390, Version 5.1 or later
  - DB2 for AS/400, Version 3.1 or later
  - DB2 for VSE & VM Version 5.1, or later.

- If you have TCP/IP network connections, you can use the TCP/IP two-phase commit support provided with DB2 Connect Enterprise Edition and DB2 Connect Personal Edition Version 5.0. This enables you to update multiple databases within a single unit of work for:

  - DB2 for OS/390, Version 5.1 or later
  - DB2 Universal Database Version 5.0 or later.

- If the application is executed under CICS for AIX or the AIX Encina Monitor, you can update multiple databases within a single unit of work for the following database products *only over SNA connections*:

  - DB2 for MVS/ESA, Version 3.1 or later
  - DB2 for OS/390, Version 5.1 or later
  - DB2 for AS/400, Version 3.1 or later
  - DB2 for VSE & VM Version 5.1, or later.

  In the TP monitor environment CICS for AIX or the Encina Monitor acts as the XA transaction manager, coordinating two-phase commits, rollbacks, and recoveries. The combination of DB2 Connect and the DRDA server can be considered an XA resource manager.

Note: TP Monitor support for two-phase commit via DB2 Connect Version 5.0 is only supported over SNA connections. There is no equivalent support for TCP/IP connections, so that - in XA environments - when DRDA servers are accessed using TCP/IP, read-only connections are all that is provided.

In order to take advantage of distributed unit of work support, you must be familiar with the concepts described in the *Administration Guide* and the SQL statements described in the *SQL Reference*.

For more information about two-phase commit, see: *Setting up Two-phase Commit Using SNA* in *DB2 Connect Enterprise Edition Quick Beginnings*, and *Setting up Two-phase Commit Using TCP/IP* in both *DB2 Connect Personal Edition Quick Beginnings* and *DB2 Connect Enterprise Edition Quick Beginnings*.

For more information about transaction monitors, and setting up a TP monitor environment, see the *CICS for AIX Customization and Operation Guide*, and *Setting up DB2 Connect with TP Monitors* in *DB2 Connect Enterprise Edition Quick Beginnings*.

DRDA and Data Access

Although DRDA defines database communication protocols, it does not define the programming interfaces, or APIs, that should be used by application programmers. In general, DRDA can be used by an application program to pass any request that a target DRDA server can execute. All of the DRDA servers available today can execute SQL requests forwarded by an application program via DB2 Connect.

IBM provides application programmers with tools to generate SQL requests for DOS, Windows, OS/2, and several UNIX platforms. These tools are part of the DB2 Software Developer Kits (DB2 SDKs). DB2 SDKs support two different API types: embedded SQL and the DB2 Call Level Interface (DB2 CLI). These can be used by programmers building applications in a variety of programming languages. For more information about these APIs, see the *Road Map to DB2 Programming*, *Building Applications for Windows and OS/2 Environments*, or *Building Applications for UNIX Environments*, as appropriate for your application platform.

Application developers can also use APIs provided by other companies. For example, the Microsoft ODBC API is used by many Windows application programmers to develop database applications. DB2 Connect provides an ODBC driver that supports applications developed using the ODBC API. IBM does not provide tools for developing ODBC applications; these tools are provided by the Microsoft Corporation.

Chapter 3. Updating Database Directories

DB2 Connect uses the following directories to manage information about databases that it connects to:

- The *node directory*, which contains network address and communication protocol information for every DRDA server that DB2 Connect accesses.
- The *database connection services* (DCS) directory, which contains information specific to DRDA server databases.
- The *system database directory*, which contains name, node, and authentication information for every database that DB2 Connect accesses.

For each DRDA server database that you connect to, you must update these directories or store equivalent information in a global DCE directory. For more information about DCE, see Appendix C, "Using DCE Directory Services" on page 91. This chapter assumes that you are *not* using DCE Directory Services.

Note: Before updating these directories, you should configure communications on the DRDA server and workstations. This is described in *DB2 Connect Personal Edition Quick Beginnings*, and *DB2 Connect Enterprise Edition Quick Beginnings*.

On OS/2, Windows 95, and Windows NT, database directories can be updated using the DB2 Universal Database Client Configuration Assistant (CCA). On Windows 3.1 and Windows for Workgroups 3.11, the database directories can be updated using the Data Sources Setup tool. On all other platforms the database directories must be updated using the DB2 Command Line Processor (CLP).

Collecting Information

Appendix A, "Directory Customization Worksheet" on page 87 shows the information that you need to collect. You may find it convenient to make a copy of the worksheet and write in the values for your system.

Node Directory

You can specify the following information in the node directory:

Node name

A nickname for the DRDA server system on which the remote database resides. This name is user-defined. Write the same node name in both the Node Directory Parameters table and the System Database Directory Parameters table.

Format: 1–8 single-byte alphanumeric characters, including the number sign (#), at sign (@), dollar sign ($), and underscore (_). It cannot begin with an underscore or a number.

Protocol

Can be APPC or TCP/IP.

Symbolic destination name

When defining an APPC node, use the symbolic destination name that was specified in the CPI Communications Side Information Table (for example, the name of the CPI-C Symbolic Destination Properties when using Microsoft SNA Server). You should get this value from the person who either installed and/or configured SNA. The symbolic destination name is case sensitive (you may encounter an SQL1338 return code).

Security type

The type of security checking that will be done. For APPC nodes, the valid options are `SAME` and `PROGRAM`. For TCP/IP nodes, `SECURITY SOCKS` is an option which specifies that the node will be SOCKS-enabled, in which case the SOCKS_NS and SOCKS_SERVER environment variables are mandatory and must be set to enable SOCKS. For more information, see Chapter 6, “Security” on page 41, and refer to the *Command Reference*.

TCP/IP remote hostname

When defining a TCP/IP node, either the remote TCP/IP hostname, or the remote TCP/IP address. If a hostname is specified, then it must be resolved at the DB2 Connect workstation, either through Domain Name Server (DNS) lookup, or by an entry in the local TCP/IP hosts file.

TCP/IP service name

When defining a TCP/IP node, either the remote TCP/IP service name or port number. This must be defined to TCP/IP at the remote host. Port number 446 has been registered as the default port number for DRDA.

Note: A second port used for two-phase commit resynchronization operations over TCP/IP connections is assigned by the server. For example, the DB2 for OS/390 bootstrap dataset assigns the port number to be used for resynchronization for inbound connections to DB2 for OS/390 only. No service name need be defined for this.

DCS Directory

You can specify the following information in the DCS directory:

Database name

A user-defined nickname for the DRDA server database. Use the same database name in both the DCS Directory Parameters table and the System Database Directory Parameters table.

Format: 1–8 single-byte alphanumeric characters, including the number sign (#), at sign (@), dollar sign ($), and underscore (_). It cannot begin with an underscore or a number.

Target database name

The database on the DRDA server system, as follows:

MVS or OS/390	The LOCATION value
VSE or VM	The database name (DBNAME)

OS/400 The relational database name (RDBNAME)
Other For OS/2 and UNIX-based systems, the database alias.

The default is the value that you specify for Database name.

Application requester

The name of the application requester that forwards SQL requests to DRDA application servers. The application requester handles requests on behalf of an application program. The default is the DB2 Connect application requester.

Parameter string

If you want to change the defaults, specify any or all the following parameters in the following order:

map-file The name of an SQLCODE mapping file that overrides the default SQLCODE mapping. To turn off SQLCODE mapping, specify **NOMAP**.

For more information, see Chapter 8, "SQLCODE Mapping" on page 61.

,D

The application will disconnect from the DRDA server database when this parameter (a comma followed by D) is used, and one of the following SQLCODES is returned:

SQL30000N
SQL30040N
SQL30050N
SQL30051N
SQL30053N
SQL30060N
SQL30070N
SQL30071N
SQL30072N
SQL30073N
SQL30074N
SQL30090N

When the disconnect parameter **,D** is not specified, a disconnect will be performed only when the following SQLCODEs are returned:

SQL30020N
SQL30021N
SQL30041N
SQL30061N
SQL30081N

For explanations of these codes, see the *Message Reference*.

Note: If DB2 Connect disconnects due to an error, a rollback will be done automatically.

,,INTERRUPT_ENABLED

If INTERRUPT_ENABLED is configured in the DCS directory at the DB2 Connect workstation, and a client application issues an interrupt while connected to the DRDA server, DB2 Connect will perform the interrupt by dropping the connection and rolling back the unit of work. This interrupt behavior is supported on AIX, OS/2, and Windows NT.

The application will receive sqlcode (-30081) indicating that the connection to the server has been terminated. The application must then establish a new connection with the DRDA server, in order to process additional database requests. Note that on platforms other than AIX V4.1 with SNA Server V3.1, OS/2, and Windows NT, DB2 Connect does not support the option of automatically disconnecting when an application using it receives an interrupt request.

Note: This support works for TCP/IP connections on any platforms. The client may kill the socket, but - depending on the server implementation - there may or may not be an outstanding receive. DB2 for OS/390 utilizes asynchronous socket calls and therefore is able to detect the loss of the connection and roll back any long-running SQL statements that are in progress.

For example, you could specify any of the following:

```
On AIX:

   NOMAP
   /u/username/sqllib/map/dcs1new.map,D
   ,D
   ,,INTERRUPT_ENABLED

On OS/2:

   NOMAP
   d:\sqllib\map\dcs1new.map,D
   ,,INTERRUPT_ENABLED
```

or accept the defaults by not specifying a parameter string.

System Database Directory

You can specify the following information in the system database directory:

Database name
The same value that you wrote in the DCS Directory Parameters table.

Database alias
An alias for the DRDA server database. This name will be used by any application program that accesses the database. By default, the value that you specify for Database name is used.

Format: 1–8 single-byte alphanumeric characters, including the number sign (#), at sign (@), dollar sign ($), and underscore (_). It cannot begin with an underscore or a number.

Node name
The same value that you wrote in the Node Directory Parameters table.

Authentication
Specifies where the validation of the user's name and password will be done. The valid options are: `SERVER`, `CLIENT`, `DCE`, and `DCS`. For more information, see Chapter 6, “Security” on page 41.

Defining Multiple Entries for the Same Database

For each database, you must define at least one entry in each of the three directories (node directory, DCS directory, and system database directory). In some cases, you might want to define more than one entry for the database.

For example, you might want to turn off SQLCODE mapping for applications that were ported from the DRDA server but accept the default mapping for applications that were developed for the client/server environment. You would do this as follows:

- Define one entry in the node directory.
- Define two entries in the DCS directory, with different database names. For one entry, specify NOMAP in the parameter string.
- Define two entries in the system database directory, with different database aliases and the two database names that you specified in the DCS directory.

Both aliases access the same database, one with SQLCODE mapping and the other without SQLCODE mapping.

Updating the Directories

You can use the command line processor CATALOG command on any DB2 Connect system; the “Add Database SmartGuide” of the Client Configuration Assistant on OS/2, Windows 95, or Windows NT; or the Data Sources Setup tool on Windows 3.1 or Windows 3.11 for Workgroups. If you have the DB2 Software Developer Kit (SDK), you can also create an application program to catalog entries. For information about APIs, see the *API Reference* and the *Command Reference*. Note that to catalog a database, you must have *sysadm* authority.

To update the directories using the command line processor, do the following:

1. Use one of the following commands to update the node directory:

   - For a node having an APPC connection:

     ```
     db2 CATALOG APPC NODE nodename
     REMOTE symbolic_destination_name
     SECURITY security_type
     ```

     For example:

     ```
     db2 CATALOG APPC NODE DB2NODE REMOTE DB2CPIC SECURITY PROGRAM
     ```

   - For a DB2 for OS/390 Version 5.1 database having a TCP/IP connection:

     ```
     db2 CATALOG TCPIP NODE nodename
     REMOTE hostname or IP address
     SERVER service_name or port_number
     SECURITY security_type
     ```

     For example:

     ```
     db2 CATALOG TCPIP NODE MVSIPNOD REMOTE MVSHOST SERVER DB2INSTC SECURITY NONE
     ```

     The default DRDA port number for TCP/IP connections is 446.

2. Use the following command to update the DCS directory:

   ```
   db2 CATALOG DCS DATABASE database_name
   AS target_database_name
   [AR application_requester]
   [PARMS "parameter string"]
   ```

   For example:

   ```
   db2 CATALOG DCS DATABASE DB2DB AS NEW_YORK3
   ```

   Or, for OS/2:

   ```
   db2 CATALOG DCS DATABASE DB2DB AS NEW_YORK3 PARMS "NOMAP,D"
   ```

   Or, for AIX:

   ```
   db2 CATALOG DCS DATABASE DB2DB AS NEW_YORK3 PARMS '"NOMAP,D"'
   ```

3. Use the following command to update the system database directory:

   ```
   db2 CATALOG DATABASE database_name
   AS alias
   AT NODE nodename
   AUTHENTICATION authentication_type
   ```

   For example:

   ```
   db2 CATALOG DATABASE DB2DB AS NYC3 AT NODE DB2NODE AUTHENTICATION DCS
   ```

If you have remote clients, you must also update directories on each remote client. For more information, see *DB2 Connect Enterprise Edition Quick Beginnings*.

Chapter 4. Binding Applications and Utilities

Application programs developed using embedded SQL must be bound to each database with which they will operate. On platforms where these functions are available, you can do this using the Command Center and the Client Configuration Assistant.

Binding should be performed once per application, for each database. During the bind process, database access plans are stored for each SQL statement that will be executed. These access plans are supplied by application developers and are contained in *bind files*, which are created during precompilation. Binding is simply a process of processing these bind files by a DRDA Server. For more information about binding, refer to the *Administration Guide*.

Because several of the utilities supplied with DB2 Connect are developed using embedded SQL, they must be bound to a DRDA server before they can be used with that system. If you do not use the DB2 Connect utilities and interfaces listed in Table 1 on page 22, you do not have to bind them to each of your DRDA servers. The lists of bind files required by these utilities are contained in the following files:

ddcsmvs.lst	For MVS or OS/390
ddcsvse.lst	For VSE
ddcsvm.lst	For VM
ddcs400.lst	For OS/400

Binding one of these lists of files to a database will bind each individual utility to that database.

If DB2 Connect Enterprise Edition (formerly DDCS Multi-user Gateway) is installed, the DB2 Connect utilities must be bound to each DRDA server; once from each type of client platform, before they can be used with that system.

For example, if you have 10 OS/2 clients, 10 Windows clients, and 10 AIX clients connecting to DB2 for OS/390 via a DB2 Connect Enterprise Edition for Window NT gateway, do the following:

1. Bind ddcsmvs.lst from one of the Windows clients.
2. Bind ddcsmvs.lst from one of the OS/2 clients.
3. Bind ddcsmvs.lst from one of the AIX clients.
4. Bind ddcsmvs.lst from the DB2 Connect gateway workstation.

Note: This assumes all the clients are at the same service level. If they are not then, in addition, you may need to bind from each client of a particular service level. Refer to Appendix D, “Binding Utilities for Back-Level Clients” on page 101 if you have clients prior to DB2 Version 2.1.

In addition to DB2 Connect utilities, any other applications that use embedded SQL must also be bound to each database that you want them to work with. An application that is not bound will usually produce an SQL0805N error message when executed. You might want to create an additional bind list file for all of your applications that need to be bound.

For each DRDA server database that you are binding to, do the following:

1. Make sure that you have sufficient authority for your DRDA server database management system:

   MVS or OS/390 The authorizations required are:

   SYSADM or
   SYSCTRL or
   BINDADD *and* CREATE IN COLLECTION NULLID

   Note: The BINDADD and the CREATE IN COLLECTION NULLID privileges provide sufficient authority **only** when the packages do not already exist. For example, if you are creating them for the first time.

   If the packages already exist, and you are binding them again, then the authority required to complete the task(s) depends on who did the original bind.

   A If you did the original bind and you are doing the bind again, then having any of the above listed authorities will allow you to complete the bind.

   B If your original bind was done by someone else and you are doing the second bind, then you will require either the SYSADM **or** the SYSCTRL authorities to complete the bind. Having just the BINDADD and the CREATE IN COLLECTION NULLID authorities will not allow you to complete the bind. It is still possible to create a package if you do not have either SYSADM or SYSCTRL privileges. In this situation you would need the BIND privilege on each of the existing packages that you intend to replace.

   VSE or VM The authorization required is DBA authority. If you want to use the GRANT option on the bind command (to avoid granting access to each DB2 Connect package individually), the NULLID user ID must have the authority to grant authority to other users on the following tables:

   system.syscatalog
   system.syscolumns
   system.sysindexes
   system.systabauth
   system.syskeycols
   system.syssynonyms
   system.syskeys
   system.syscolauth

   On the VSE or VM system, you can issue:

   ```
   grant select on table to nullid with grant option
   ```

OS/400 *CHANGE authority or higher on the NULLID collection.

2. Issue commands similar to the following:

   ```
   db2 connect to DBALIAS user USERID using PASSWORD
   db2 bind path@ddcsmvs.lst blocking all
         sqlerror continue messages ddcsmvs.msg grant public
   db2 connect reset
   ```

   Where *DBALIAS*, *USERID*, and *PASSWORD* apply to the DRDA server database, ddcsmvs.lst is the bind list file for MVS, and *path* is the location of the bind list file.

   For example `drive:\sqllib\bnd\` applies to all Intel operating systems, and `INSTHOME/sqllib/bnd/` applies to all UNIX operating systems, where *drive* is the logical drive where DB2 Connect was installed and *INSTHOME* is the home directory of the DB2 Connect instance.

   You can use the `grant` option of the `bind` command to grant EXECUTE privilege to PUBLIC or to a specified user name or group ID. If you do not use the `grant` option of the `bind` command, you must GRANT EXECUTE (RUN) individually.

   To find out the package names for the bind files, enter the following command:

   ```
   ddcspkgn @bindfile.lst
   ```

   For example:

   ```
   ddcspkgn @ddcsmvs.lst
   ```

   might yield the following output:

   ```
   Bind File                      Package Name
   ------------------------------ ------------------------------
   f:\sqllib\bnd\db2ajgrt.bnd     SQLAB6D3
   ```

   For your reference, Table 1 on page 22 shows the bind files and package names that are used by different components of DB2 Connect. In some cases, different bind files and packages are used on different operating systems.

Table 1. Bind Files and Packages

Component	Bind file	Package	MVS or OS/390	VSE	VM	OS/400
Binder (used by the GRANT bind option)	db2ajgrt.bnd	sqlab*xxx*	yes	yes	yes	yes
DB2 Call Level Interface						
Isolation level CS	db2clics.bnd	sqll1*xxx*	yes	yes	yes	yes
Isolation level RR	db2clirr.bnd	sqll2*xxx*	yes	yes	yes	yes
Isolation level UR	db2cliur.bnd	sqll3*xxx*	yes	yes	yes	yes
Isolation level RS	db2clirs.bnd	sqll4*xxx*	yes	yes	yes	yes
Isolation level NC	db2clinc.bnd	sqll5*xxx*	no	no	no	yes
Using MVS table names	db2clims.bnd	sqll7*xxx*	yes	no	no	no
Using OS/400 table names (OS/400 3.1 or later)	db2clias.bnd	sqlla*xxx*	no	no	no	yes
Using VSE/VM table names	db2clivm.bnd	sqll8*xxx*	no	yes	yes	no
Command Line Processor						
Isolation level CS	db2clpcs.bnd	sqlc2*xxx*	yes	yes	yes	yes
Isolation level RR	db2clprr.bnd	sqlc3*xxx*	yes	yes	yes	yes
Isolation level UR	db2clpur.bnd	sqlc4*xxx*	yes	yes	yes	yes
Isolation level RS	db2clprs.bnd	sqlc5*xxx*	yes	yes	yes	yes
Isolation level NC	db2clpnc.bnd	sqlc6*xxx*	no	no	no	yes
REXX						
Isolation level CS	db2arxcs.bnd	sqla1*xxx*	yes	yes	yes	yes
Isolation level RR	db2arxrr.bnd	sqla2*xxx*	yes	yes	yes	yes
Isolation level UR	db2arxur.bnd	sqla3*xxx*	yes	yes	yes	yes
Isolation level RS	db2arxrs.bnd	sqla4*xxx*	yes	yes	yes	yes
Isolation level NC	db2arxnc.bnd	sqla5*xxx*	no	no	no	yes
Utilities						
Export	db2uexpm.bnd	sqlub*xxx*	yes	yes	yes	yes
Import	db2uimpm.bnd	sqluf*xxx*	yes	yes	yes	yes

where *xxx* depends on the type of client platform and what service you have applied.

The following are the respective xxx values:

4C0	DB2 Client Application Enabler for AIX Version 2.1
4D0	DB2 Client Application Enabler for OS/2 Version 2.1
4W0	DB2 Client Application Enabler for Windows Version 2.1
6A3	DB2 Client Application Enabler for AIX
6D3	DB2 Client Application Enabler for OS/2 Version 2.1.2
6H3	DB2 Client Application Enabler for HP-UX
6G3	DB2 Client Application Enabler for Silicon Graphics Version 2.1
6M2	DB2 Client Application Enabler for Macintosh Version 2.1.2
6N1	DB2 Client Application Enabler for Windows 95 and NT Version 2.
6P1	DB2 Client Application Enabler for Windows NT (PPC) Version 2.1
6S1	DB2 Client Application Enabler for SCO OpenServer Version 2.1.2
6U3	DB2 Client Application Enabler for Solaris
6W4	DB2 Client Application Enabler for Windows Version 2.1.2
6X3	DB2 Client Application Enabler for SINIX
7C0	DB2 Universal Database Version 5

For the most recent version of this information, refer to *Installing and Configuring DB2 Clients*. To determine these values for DB2 Connect V5.0 execute the *ddcspkgn* utility, for example:

```
ddcspkgn @ddcsmvs.lst
```

Optionally, this utility can be used to determine the package name of individual bind files, for example:

```
ddcspkgn bindfile.bnd
```

If your DB2 for MVS/ESA system has the fix for APAR PN60988 installed (or if it is a later release than Version 3 Release 1), you can also add the bind files for isolation level NC to the *ddcsmvs.lst file*.

Bind options are discussed in detail in the *Command Reference*.

Notes:

a. Using the bind option `sqlerror continue` is not optional. Specifying this option turns bind errors into warnings, so that binding a file containing errors can still result in the creation of a package. In turn, this allows one bind file to be used against multiple servers even when a particular server implementation may flag the SQL syntax of another to be invalid. For this reason, binding any of the list files `ddcsxxx.lst` against any particular DRDA server should be expected to produce some warnings. For example, when binding against DB2 for VM, numerous warning messages may result since DB2 for VM does not permit cursors to be declared as `"WITH HOLD"`.

b. If you are connecting to a DB2 Universal Database database through DB2 Connect, use the bind list `db2ubind.lst` and do not specify `sqlerror continue`, which is only valid when connecting to a DRDA server. Also, to connect to a DB2 Universal Database database, we recommend that you use the Client Application Enablers provided with DB2 and not DB2 Connect.

3. Use similar statements to bind each application or list of applications.

4. If you have remote clients from a previous release of DB2, you may need to bind the utilities on these clients to DB2 Connect. See Appendix D, “Binding Utilities for Back-Level Clients” on page 101 for more information.

The BIND Command

The DB2 **bind** command binds an application to a specific database. If you precompile and bind in separate operations, the options that you specify in the bind will override the options that you specified in the precompile step.

The *Command Reference* describes the syntax of the BIND command that you must use when binding an application to a DRDA server through DB2 Connect. Be sure to check that you are referring to the DRDA-specific description.

Note: Some parameters of the BIND command may not be supported by your DRDA server. For more information, see the documentation supplied with your DRDA server RDBMS.

Rebinding

After you bind your application (and create the package on the DRDA server), you may find that you need to recreate that package. It is possible to do this without having the original bind file, by using the command line processor **REBIND PACKAGE** command or corresponding API.

The benefits of using this command are:

- You can take advantage of changes in the system by re-optimizing and building new package sections without having the original bind file.
- You can recreate packages that have been made inoperative or invalid.
- You can recreate packages that have been invalidated by migration.
- You can improve performance by using an explicit rebind rather than using an implicit rebind or a bind.
- You can change characteristics. For example, with DB2 for OS/390, you can change the qualifier of unqualified tables for testing or migration purposes.

If you want to modify a program, the bind options, or any owner information, you should use the **BIND** command. Also, if the package does not exist in the database, or if you want to see all bind errors (not only the first detected error), you should use the **BIND** command.

To run this command, you need the level of authority required by your DRDA server. If you are not connected to a database, the command will cause an implicit connect to be done to the default database (if you have connect privileges).

The syntax of the command line processor command is described in the *Command Reference*.

To find out the package name for a bind file, enter the following command:

```
ddcspkgn bindfile.bnd
```

Chapter 5. DB2 Connect Administration Utilities

This chapter describes utilities that help you perform administration tasks. It contains the following sections:

- "Command Line Processor"
- "Database System Monitor"
- "Using Import and Export Utilities" on page 29
- "Trace Utility (ddcstrc)" on page 30.

Command Line Processor

The command line processor lets you issue SQL statements against a DRDA server database, preceded by **db2**. For differences between DRDA server SQL and DB2 Connect SQL, see "DRDA Server SQL Statements Supported by DB2 Connect" on page 57 and "DRDA Server SQL Statements Rejected by DB2 Connect" on page 57.

You can also enter **db2** with no SQL statement. This puts you into interactive input mode, where you can type SQL commands without prefixing them by **db2**. Enter **quit** to leave CLP interactive input mode, or enter **terminate** to end the command line processor and the database connection.

Before you can use the command line processor, it must be bound, as described in Chapter 4, "Binding Applications and Utilities" on page 19.

Note: When using the command line processor in UNIX-based systems without being in interactive input mode, you must put double quotes around special characters (such as * and ?) when issuing SQL statements.

For more information on SQL commands, see the *SQL Reference*.

Database System Monitor

You can use the Database System Monitor to monitor the connections for remote clients.

When an error occurs at the DRDA server, the system administrator can determine if the problem was on the DB2 Connect workstation. The database system monitor correlates:

- The DRDA correlation token (CRRTKN), for unprotected conversations.
- The logical unit of work identifier (LUWID), for two-phase conversations protected by an SNA Syncpoint Manager (SPM).
- The unit of work id (UOWID), for two-phase connections protected by the DRDA-3 Syncpoint Manager (as used over TCP/IP connections).
- The DB2 Connect connection identifier (the Application ID).

This shows which DB2 Connect connection caused the problem, which allows the system administrator to force the individual client application from the system without affecting the other clients using the DB2 Connect connection.

For detailed information about the database system monitor and how it works, see the *System Monitor Guide and Reference*.

To activate the monitor, use the DB2 UPDATE MONITOR SWITCHES command. Refer to *Command Reference* for the syntax of this command. Here is an example, which creates DB2 System Monitor events for Units of Work (UOW):

```
db2 update monitor switches using uow on
```

To list the status of monitor switches, use the DB2 GET MONITOR SWITCHES command.

To view the information provided by the monitor at the application level, issue the DB2 LIST DCS APPLICATIONS command. It returns the following information for an APPC connection (DB2 Connect Enterprise Edition Version 5.0 to DB2 for OS/390):

```
Auth Id  Application Name     Appl.      Outbound Application Id
                            Handle
-------- -------------------- ---------- --------------------------------
USERID   db2bp_41             0          CAIBMOML.OMXT4H0A.A79EAA3C6E29
```

It returns the following information for a TCP/IP connection (DB2 Connect Enterprise Edition Version 5.0 to DB2 for OS/390):

```
Auth Id  Application Name     Appl.      Outbound Application Id
                              Handle
-------- -------------------- ---------- --------------------------------
USERID   db2bp_41             2          0915155C.9704.1517172201BE
```

Auth.Id
The authorization ID that was used to log on to the DRDA server. This identifies who is running the application.

Application Name
The name of the application running at the client as known to DB2 Connect. Only the first 20 bytes after the last path separator are available. The application name is not available for applications running on DB2 for OS/2 Version 1.

Appl. Handle
The agent that is executing on the DB2 Connect workstation. You can use this element to link the database system monitor information to other diagnostic information. (For example, see “Trace Utility (ddcstrc)” on page 30.) The agent ID is also required when using the FORCE USERS command or API.

Outbound Application ID
One of the following:

- The DRDA correlation token (CRRTKN), for unprotected conversations.
- The logical unit of work identifier (LUWID), for two-phase conversations protected by an SNA Syncpoint Manager (SPM).
- The unit of work id (UOWID), for two-phase connections protected by the DRDA-3 Syncpoint Manager (as used over TCP/IP connections).

This unique identifier is generated when the application connects to the DRDA server database. You can use this element in conjunction with the Application ID to correlate the client and server parts of the application information.

If the DB2 LIST DCS APPLICATIONS SHOW DETAIL command format is specified, additional information is shown, including, for an APPC connection:

```
Auth Id  Application Name     Appl.      Client Application Id
                              Handle
-------- -------------------- ---------- --------------------------------
USERID   db2bp_41             0          09150FDB.9704.1517161810D5

Seq# DB Alias Node     Client   Codepage   Outbound Application Id
---- -------- -------- -------- ---------- --------------------------------
0001 DB2SF    weasel   SQL05000 850        CAIBMOML.OMXT4H0A.A79EAA3C6E29

Seq# DB Name              Host
---- -------------------- --------
0000 SAN_FRANCISCO        DSN05010
```

Or, for a TCP/IP connection:

```
Auth Id  Application Name     Appl.      Client Application Id
                              Handle
-------- -------------------- ---------- --------------------------------
USERID   db2bp_41             2          09150FDB.9704.1517172110D6

Seq# DB Alias Node     Client   Codepage   Outbound Application Id
---- -------- -------- -------- ---------- --------------------------------
0001 DB2SFT   weasel   SQL05000 850        0915155C.9704.1517172201BE

Seq# DB Name              Host
---- -------------------- --------
0000 SAN_FRANCISCO        DSN05010
```

Client Application ID

Uniquely identifies the application connected to the DB2 Connect workstation. There are different formats for the application ID, which are dependent on the communication protocol between the client and the DB2 Connect workstation. For more information on the formats, see the *Administration Guide*.

This value lets you correlate connections from clients to the DB2 Connect workstation and from the DB2 Connect workstation to the DRDA server.

Client Sequence no (Seq#)

The client sequence number is the transaction sequence number. It is used to help correlate a transaction spread over different systems.

Client DB alias

The alias of the database provided by the application to connect to the database. This element can be used to identify the actual database that the application is accessing. The mapping between this name and the database name could be done by using the database directories at the client node and the database manager server node.

Client NNAME (Node)
Identifies the node where the client application is executing. The information varies according to the client protocol in use. For example, for a client connected via NetBIOS, this is the value of the NNAME database manager configuration parameter. For a client connected via TCP/IP, this is the host name.

Client Product ID (Client)
The product and version that is running on the client. The client product IDs will be:

- SQL01010 for Version 1 of DB2 for OS/2
- SQL01011 for Version 1 of UNIX-based DB2 products and Client Application Enablers.
- SQL02010 for Version 2 of DB2 products and Client Application Enablers.
- SQL02020 for Version 2.1.2 of DB2 products and Client Application Enablers.
- SQL05000 for Version 5.0 of DB2 Universal Database and DB2 Connect products and their clients.

Code Page ID
The code page identifier at the node where the monitored application started.

You can use this information to ensure that data conversion is supported between the application code page and the database code page (or for DRDA server databases, the DRDA server CCSID).

If the application code page is different from that under which the database system monitor is running, this code page element can help you to manually convert data that was passed from the application and displayed by the database system monitor. For example, you can use it to help translate the Application Name.

Outbound Sequence No
This represents the outbound sequence number. It is used for correlating transactions on different systems.

Host Database Name
The real name of the database to which the application is connected. In the DCS directory, this is the *target database name*.

Host Product ID
The product and version that is running on the server. It is in the form *PPPVVRRM*, where:

PPP Identifies the DRDA server product (for example, DSN for DB2 for OS/390, ARI for DB2 for VSE & VM, or QSQ for DB2 for AS/400)
VV Is a two-digit version number, such as `01`
RR Is a two-digit release number, such as `01`
M Is a one-digit modification level.

Using Import and Export Utilities

The import and export utilities let you move data from a DRDA server database to a file on the DB2 Connect workstation or vice versa. You can then use this data with any other application or RDBMS that supports this import/export format. For example, you can export data from DB2 for MVS/ESA into a delimited ASCII file and later import it into a DB2 for OS/2 database.

You can perform export and import functions from a database client or from the DB2 Connect workstation.

Notes:

1. The data to be imported or exported must comply with the size and data type restrictions of both databases.
2. To improve import performance, you can use compound SQL. Specify `COMPOUND=`*`number`* in the import API or the CLP *filetype-mod* string parameter to group the specified number of SQL statements into a block. This may reduce network overhead and improve response time.
3. For information on the syntax of the import and export utilities from the command line processor, see the *Command Reference* manual.

Moving Data from a Workstation to a DRDA Server

To export to a DRDA server database:

1. Export the rows of information from the DB2 table into a PC/IXF file.
2. If the DRDA server database does not contain a table having attributes compatible with the information to be imported into it, create a compatible table.
3. Using the INSERT option, import the PC/IXF file to a table in the DRDA server database.

Moving Data from a DRDA Server to a Workstation

To import data from a DRDA server database:

1. Export the rows of information from the DRDA server database table to a PC/IXF file.
2. Use the PC/IXF file for importing to a DB2 table.

Restrictions

With the DB2 Connect program, import or export operations must meet the following conditions:

- The file type must be PC/IXF.
- Index definitions are not stored on export or used on import.
- A table with attributes that are compatible with those of the data must exist before you can import to it. Importing through the DB2 Connect program cannot create a table because INSERT is the only supported option.

- A commit count interval must not be specified with import.

If these conditions are violated, the operation will fail and an error message will be generated.

Mixed Single-Byte and Double-Byte Data

If you import and export mixed data (columns containing both single-byte and double-byte data), consider the following:

- On systems that store data in EBCDIC (MVS, OS/390, OS/400, VM, and VSE), shift-out and shift-in characters mark the start and end of double-byte data. When you define column lengths for your database tables, be sure to allow enough room for these characters.
- Variable-length character columns are recommended unless the data in a column has a consistent pattern. If it does, fixed length is acceptable.

Replacement for SQLQMF Utility

The function of the SQLQMF utility with DDCS for OS/2 has been replaced by the DB2 Connect Import/Export functions. The advantages are:

- No need for QMF on the host
- No need to logon to the host (a TSO id is still required on DB2 for MVS/ESA or DB2 for OS/390)
- Supports DB2 for MVS, DB2 for OS/390, DB2 for OS/400, and DB2 for VM and VSE
- Good performance achieved by using compound SQL
- Supports several file formats, in addition to ASCII
- Can be run from a client machine with no SNA connectivity.

Refer to the *Command Reference* for further information on using these commands.

Trace Utility (ddcstrc)

The **ddcstrc** utility provides a record of the data interchanged between the DB2 Connect workstation (on behalf of the database client) and the DRDA server database management system.

As a database administrator (or application developer), you may find it useful to understand how this flow of data works, because this knowledge can help you determine the origin of a particular problem. For example, say you issue a `CONNECT TO` database statement for a DRDA server database, but the command fails and you receive an unsuccessful return code. If you understand exactly what information was conveyed to the DRDA server database management system, you may be able to determine the cause of the failure even if the return code information is general. Many failures are caused by simple user errors.

Output from `ddcstrc` lists the data streams exchanged between the DB2 Connect workstation and the DRDA server database management system. Data sent to the DRDA server is labeled SEND BUFFER and data received from the DRDA server is labeled RECEIVE BUFFER. If a receive buffer contains SQLCA information, it will be followed by a formatted interpretation of this data and labeled SQLCA. The SQLCODE field of an SQLCA is the *unmapped* value as returned by the DRDA server system. (For more on mapping, see Chapter 8, "SQLCODE Mapping" on page 61.) The send and receive buffers are arranged from the oldest to the most recent within the file. Each buffer has:

- The process ID
- A SEND BUFFER, RECEIVE BUFFER, or SQLCA label. The first DDM command or object in a buffer is labeled DSS TYPE.

The remaining data in send and receive buffers is divided into five columns, consisting of:

- A byte count.
- Columns 2 and 3 represent the DRDA data stream exchanged between the two systems, in ASCII or EBCDIC.
- An ASCII representation of columns 2 and 3.
- An EBCDIC representation of columns 2 and 3.

For more information about DDM, see:

- *DB2 for OS/390 Reference for Remote DRDA Requesters and Servers*
- *Distributed Relational Database Reference*
- *Distributed Data Management Architecture Level 3: Reference*

Trace Syntax

This command is invoked from the operating system command prompt with the following syntax

```
                               ┌───────────┐
►►─┬──────────┬─ddcstrc───on──▼─┬─-r─┬─┴──┬───────────────┬────────────────────►
   └─path─┘                     ├─-s─┤     └─-l=length─┘
                                └─-c─┘

►─off─┬──────────────────┬──┬──────────┬──────────────────────────────────────►◄
       └─-t=tracefile─┘  └─-p=pid─┘
```

Figure 3. Syntax of the ddcstrc Command

Note: The syntax of this command may vary slightly depending on the operating system that you are using. For example, ***/*** may be used in place of **-** for the OS/2 operating system.

Trace Parameters

on Turns on DRDA application requester trace events.

off Turns off DRDA application requester trace events.

-r Traces DRDA data streams received from the DRDA server system.

-s Traces DRDA data streams sent to the DRDA server system.

-c Traces the SQLCA received from the DRDA server system.

The default is -r, -s, and -c.

-l=*length* Specifies the size of the buffer used to store the trace information. The default is 1M, and the minimum is 64K.

-t=*tracefile* Specifies the destination for the trace; *tracefile* may be the name of a file or a standard device. If a file name is specified without a complete path, the current path is used for the missing parts. The default file name is `ddcstrc.dmp`.

-p=*pid* Traces events only for this process. If -p is not specified, all processes for the user's instance are written to the output file.

Note: For a remote client, the *pid* can be found in the Agent ID field returned by the database system monitor.

For more information, see “Database System Monitor” on page 25.

Trace Output

ddcstrc writes the following information to *tracefile*:

-r
- Type of DRDA reply/object
- Receive buffer

-s
- Type of DRDA request
- Send buffer

-c
- SQLCA

CPI-C error information
- Receive function return code
- Severity
- Protocol used
- API used
- Function
- CPI-C return code
- Error number
- Internal return code.

SNA error information
- Receive function return code
- Severity
- Protocol used

- Function
- Partner LU name
- Error number.

TCP/IP error information

- Receive function return code
- Severity
- Protocol used
- API used
- Function
- Error number.

Note:

1. A value of zero for the exit code indicates that the command completed successfully, and a non-zero value indicates that it did not.
2. The fields returned vary based on the API used. The SNA API is used only for 2PC SPM connections.
3. The fields returned vary based on the platform on which DB2 Connect is running, even for the same API.
4. If ddcstrc sends the output to a file that already exists, the old file will be erased unless the permissions on the file do not allow it to be erased.

Analyzing the Trace Output File

The following pages show example output illustrating some DRDA data streams exchanged between DB2 Connect workstations and a DRDA server. From the user's viewpoint, a `CONNECT TO` database command has been issued using the command line processor.

Figure 4 on page 34 uses DB2 Connect Enterprise Edition Version 5 and DB2 for OS/390 Version 5.1 over an APPC connection.

Figure 5 on page 36 uses DB2 Connect Enterprise Edition Version 5 and DB2 for OS/390 Version 5.1 over a TCP/IP connection.

```
1       DB2 fnc_data      gateway_drda_ar      sqljcsend (1.35.10.80)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 177

        SEND BUFFER:  EXCSAT RQSDSS         (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  006AD04100010064 10410020115E8482   .j.A...d.A. .¬..  .|}..........;db
  0010  F282976DF3F24040 4040404040404040   ...m..@@@@@@@@@@  2bp_32
  0020  4040F0F0F0F0C2C5 F5C3000C116DA685   @@...........m..    0000BE5C..._we
  0030  81A2859340400013 115AC4C2F240C396   ....@@...Z...@..  asel  ..."DB2 Co
  0040  95958583A340F54B F000141404140300   .....@.K........  nnect 5.0.......
  0050  0414440003240700 05240F0003000D11   ..D..$...$......  ................
  0060  47D8C4C2F261F6F0 F0F000A0D0010002   G....a..........  .QDB2/6000..}...
  0070  009A200100162110 E2C1D56DC6D9C1D5   .. ...!....m....  ........SAN_FRAN
  0080  C3C9E2C3D6404040 40400006210F2407   .....@@@@@..!.$.  CISCO     ......
  0090  000D002FD8E3C4E2 D8D3C1E2C3000C11   .../............  ....QTDSQLASC...
  00A0  2EE2D8D3F0F5F0F0 F0003C210437E2D8   ..........<!.7..  .SQL05000.....SQ
  00B0  D3F0F5F0F0F0C1C9 E740404040404040   .........@@@@@@@  L05000AIX
  00C0  4040404040404040 8482F282976DF3F2   @@@@@@@@.....m..          db2bp_32
  00D0  4040404040404040 40404040848483A2   @@@@@@@@@@@@....              ddcs
  00E0  A4A2F14000001B21 35C3C1C9C2D4D6D4   ...@...!5.......  us1 .....CAIBMOM
  00F0  D34BD6D4E7E3F4C8 F0C1960624184511   .K..........$.E.  L.OMXT4H0Ao.....
  0100  000A00350006119C 0352               ...5.....R        ..........

2       DB2 fnc_data      gateway_drda_ar      sqljcrecv (1.35.10.81)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 178

        RECEIVE BUFFER:  EXCSATRD OBJDSS    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0056D04300010050 14430010115EC4C2   .V.C...P.C...¬..  ..}....&.....;DB
  0010  C1C1F0F6C4C3F0F8 C6F8001414041403   ................  AA06DC08F8......
  0020  0004144400032407 0005240F00030008   ...D..$...$.....  ................
  0030  1147D8C4C2F20014 116DE2C1D56DC6D9   .G.......m...m..  ..QDB2..._SAN_FR
  0040  C1D5C3C9E2C3D640 4040000C115AC4E2   .......@@@...Z..  ANCISCO   ..."DS
  0050  D5F0F5F0F1F00060 D0020002005A2201   ....... .....Z".  N05010.-}...."..
  0060  000611490000000D 002FD8E3C4E2D8D3   ...I...../......  ..........QTDSQL
  0070  F3F7F0000C112EC4 E2D5F0F5F0F1F000   ................  370....DSN05010.
  0080  0A00350006119C01 F4002D244E000624   ..5.......-$N..$  ........4...+...
  0090  4C00010023244D00 06244FFFFF001911   L...#$M..$O.....  <.....(...!.....
  00A0  E9C3C1C9C2D4D6D4 D34BE2C6D3E44040   .........K....@@  ZCAIBMOML.SFLU
  00B0  404007F6C4C2                        @@....              .6DB
```

Figure 4 (Part 1 of 2). Example of Trace Output (APPC connection)

```
3       DB2 fnc_data      gateway_drda_ar      sqljcsend (1.35.10.80)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 177
        SEND BUFFER:  RDBCMM  RQSDSS        (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000AD00100010004 200E               ........ .        ..}.......

4       DB2 fnc_data      gateway_drda_ar      sqljcrecv (1.35.10.81)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 178

        RECEIVE BUFFER:  ENDUOWRM RPYDSS    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  002BD05200010025 220C000611490004   .+.R...%"....I..  ..}.............
  0010  00162110E2C1D56D C6D9C1D5C3C9E2C3   ..!....m........  ....SAN_FRANCISC
  0020  D640404040400005 211501000BD00300   .@@@@@..!.......  O     .......}..
  0030  0100052408FF                        ...$..            ......
5       DB2 fnc_data      gateway_drda_ar      sqljmsca (1.35.10.108)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 179
        SQLCA

        SQLCAID:  SQLCA
        SQLCABC:  136
        SQLCODE:  0
        SQLERRML: 0
        SQLERRMC:
        SQLERRP:  DSN
        SQLERRDﬄ0->5": 00000000, 00000000, 00000000, 00000000, 00000000, 00000000
        SQLWARN(0->A):  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,
        SQLSTATE: 00000
```

Figure 4 (Part 2 of 2). Example of Trace Output (APPC connection)

```
1       DB2 fnc_data      gateway_drda_ar      sqljcsend (1.35.10.80)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 177

        SEND BUFFER:  EXCSAT RQSDSS         (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  006ED04100010068 10410020115E8482   .n.A...h.A. .¬..  .>}..........;db
  0010  F282976DF3F24040 4040404040404040   ...m..@@@@@@@@@@  2bp_32
  0020  4040F0F0F0F0C2C5 F5C3000C116DA685   @@..........m..    0000BE5C..._we
  0030  81A2859340400013 115AC4C2F240C396   ....@@...Z...@..  asel  ..."DB2 Co
  0040  95958583A340F54B F000181404140300   .....@.K........  nnect 5.0.......
  0050  0514740005240700 05240F0003144000   ..t..$...$....@.  .............. .
  0060  05000D1147D8C4C2 F261F6F0F0F00010   ....G....a......  .....QDB2/6000..
  0070  D0410002000A106D 000611A20003003C   .A.....m.......<  }......_...s....
  0080  D04100030036106E 000611A200030016   .A...6.n........  }......>...s....
  0090  2110E2C1D56DC6D9 C1D5C3C9E2C3D640   !....m.........@  ..SAN_FRANCISCO
  00A0  40404040000C11A1 9781A2A2A6969984   @@@@............      ....password
  00B0  000A11A0A4A28599 8984009CD0010004   ................  ....userid..}...
  00C0  0096200100162110 E2C1D56DC6D9C1D5   .. ...!....m....  .o......SAN_FRAN
  00D0  C3C9E2C3D6404040 40400006210F2407   .....@@@@@..!.$.  CISCO     ......
  00E0  000D002FD8E3C4E2 D8D3C1E2C3000C11   .../............  ....QTDSQLASC...
  00F0  2EE2D8D3F0F5F0F0 F0003C210437E2D8   ..........<!.7..  .SQL05000.....SQ
  0100  D3F0F5F0F0F0C1C9 E740404040404040   .........@@@@@@@  L05000AIX
  0110  4040404040404040 8482F282976DF3F2   @@@@@@@@.....m..          db2bp_32
  0120  4040404040404040 40404040A4A28599   @@@@@@@@@@@@....              user
  0130  8984404000001721 35F0F9F1F5F1F5F5   ..@@...!5.......  id  .....0915155
  0140  C34BF9F7F0F42312 322001BE000A0035   .K....#.2 .....5  C.9704..........
  0150  0006119C0352                        .....R            ......

2       DB2 fnc_data      gateway_drda_ar      sqljcrecv (1.35.10.81)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 178

        RECEIVE BUFFER:  EXCSATRD OBJDSS    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  005AD04300010054 14430010115EC4C2   .Z.C...T.C...¬..  ."}..........;DB
  0010  C1C1F0F6C4C3F0F8 C6F8001814041403   ................  AA06DC08F8......
  0020  0004147400052407 0005240F00031440   ...t..$...$....@  ..............
  0030  000500081147D8C4 C2F20014116DE2C1   .....G.......m..  ......QDB2..._SA
  0040  D56DC6D9C1D5C3C9 E2C3D64040400000C  .m.........@@@..  N_FRANCISCO   ..
  0050  115AC4E2D5F0F5F0 F1F00010D0430002   .Z...........C..  ."DSN05010..}...
  0060  000A14AC000611A2 00030015D0420003   ..............B..  .......s....}...
  0070  000F121900061149 0000000511A40000   .......I........  ..............u..
  0080  51D0020004004B22 0100061149000000   Q.....K"....I...  .}..............
  0090  0D002FD8E3C4E2D8 D3F3F7F0000C112E   ../.............  ...QTDSQL370....
  00A0  C4E2D5F0F5F0F1F0 000A00350006119C   ...........5....  DSN05010........
  00B0  01F4001E244E0006 244C00010014244D   ....$N..$L....$M  .4...+...<.....(
  00C0  0006244FFFFF000A 11E80915155C01BE   ..$O.........\..  ...!.....Y...*..
```

Figure 5 (Part 1 of 2). Example of Trace Output (TCP/IP connection)

```
3       DB2 fnc_data      gateway_drda_ar      sqljcsend (1.35.10.80)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 177

        SEND BUFFER:  RDBCMM  RQSDSS        (ASCII)          (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000AD00100010004 200E               ........ .        ..}.......

4       DB2 fnc_data      gateway_drda_ar      sqljcrecv (1.35.10.81)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 178

        RECEIVE BUFFER:  ENDUOWRM RPYDSS    (ASCII)          (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  002BD05200010025 220C000611490004   .+.R...%"....I..  ..}.............
  0010  00162110E2C1D56D C6D9C1D5C3C9E2C3   ..!....m........  ....SAN_FRANCISC
  0020  D640404040400005 211501000BD00300   .@@@@@..!.......  O     .......}..
  0030  0100052408FF                        ...$..            ......

5       DB2 fnc_data      gateway_drda_ar      sqljmsca (1.35.10.108)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 179
        SQLCA

        SQLCAID:  SQLCA
        SQLCABC:  136
        SQLCODE:  0
        SQLERRML: 0
        SQLERRMC:
        SQLERRP:  DSN
        SQLERRDffl0->5": 00000000, 00000000, 00000000, 00000000, 00000000, 00000000
        SQLWARN(0->A):  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,
        SQLSTATE: 00000
```

Figure 5 (Part 2 of 2). Example of Trace Output (TCP/IP connection)

The following information is captured in the traces:

- The process ID (PID) of the client application
- The RDB_NAME cataloged in the database connection services (DCS) directory
- The DB2 Connect CCSID(s)
- The DRDA server CCSID(s)
- The DRDA server database management system with which the DB2 Connect system is communicating.

The first buffer contains the Exchange Server Attributes (EXCSAT) and Access RDB (ACCRDB) commands sent to the DRDA server database management system. It sends them as a result of a `CONNECT TO` database command.

The next buffer contains the reply that DB2 Connect received from the DRDA server database management system. It contains an Exchange Server Attributes Reply Data (EXCSATRD) and an Access RDB Reply Message (ACCRDBRM).

Analyzing EXCSAT and ACCRDB

The EXCSAT command contains the workstation name of the client specified by the Server Name (SRVNAM) object, which is code point X'116D', according to DDM specification. The EXCSAT command is found in the first buffer. Within the EXCSAT command, the values X'116DA68581A28593' (coded in CCSID 500) are translated to *weasel* once the X'116D' is removed.

The EXCSAT command also contains the EXTNAM (External Name) object, which is often placed in diagnostic information on the DRDA server database management system. It consists of a 20-byte application ID followed by an 8-byte process ID (or 4-byte process ID and 4-byte thread ID). It is represented by code point X'115E', and in this example its value is db2bp_32 padded with blanks followed by 0000BE5C. On a UNIX-based database client, this value can be correlated with the **ps** command, which returns process status information about active processes to standard output.

The ACCRDB command contains the RDB_NAME in the RDBNAM object, which is code point X'2110'. The ACCRDB command follows the EXCSAT command in the first buffer. Within the ACCRDB command, the values X'2110E2C1D56DC6D9C1D5C3C9E2C3D6' are translated to SAN_FRANCISCO once the X'2110' is removed. This corresponds to the target database name field in the DCS directory.

The accounting string (described in “Implementing Chargeback Accounting” on page 58) has code point X'2104'.

The code set configured for the DB2 Connect workstation is shown by locating the CCSID object CCSIDSBC (CCSID for single-byte characters) with code point X'119C' in the ACCRDB command. In this example, the CCSIDSBC is X'0352', which is 850.

If the additional objects CCSIDDBC (CCSID for double-byte characters) and CCSIDMBC (CCSID for mixed-byte characters), with code points X'119D' and X'119E' respectively, are present, the DB2 Connect workstation is configured for DBCS code page support. Since the example output file does not include the two additional code points, the workstation is not configured for DBCS.

Note: The TCP/IP flows contains two new commands: ACCSEC used to access the security manager and exchange supported security mechanisms, and SECCHK, which contains the authentication tokens used to authenticate the end user of the connection. ACCSEC and SECCHK only appear for TCP/IP connections, and they do so between EXCSAT and ACCRDB.

Analyzing EXCSAT and ACCRDBRM

CCSID values are also returned from the DRDA server in the Access RDB Reply Message (ACCRDBRM) within the second buffer. This buffer contains the EXCSATRD followed by the ACCRDBRM. The example output file contains CCSID values for the DRDA server system of 500 (X'01F4', SBCS CCSID).

If DB2 Connect does not recognize the code page coming back from the DRDA server, SQLCODE -332 will be returned to the user with the source and target code pages. If the DRDA server doesn't recognize the code set sent from DB2 Connect, it will return VALNSPRM (Parameter Value Not Supported, with DDM code point `X'1252'`), which gets translated into SQLCODE -30073 for the user.

The ACCRDBRM also contains the parameter PRDID (Product-specific Identifier, with code point `X'112E'`). The value is `X'C4E2D5F0F5F0F1F0'`. This hex string corresponds to DSN05010 in EBCDIC. According to standards, DSN is DB2 for MVS/ESA or DB2 for OS/390. The version, 5.1, is also indicated. ARI is DB2 for VSE & VM, SQL is DB2 Common Server, and QSQ is DB2 for AS/400.

Analyzing Subsequent Buffers

You can analyze subsequent send and receive buffers for additional information. The third buffer contains a commit. The **commit** command instructs the DRDA server database management system to commit the current unit of work. The fourth buffer is received from the DRDA server database management system as a result of a commit or rollback. It contains the End Unit of Work Reply Message (ENDUOWRM), which indicates that the current unit of work has ended. In this example, it contains a null SQLCA, indicated by DDM code point `X'2408'` followed by `X'FF'`. A null SQLCA (`X'2408FF'`) indicates success (SQLCODE 0). When a receive buffer contains an SQLCA (possibly a null SQLCA), ddcstrc will follow this receive buffer with a formatted interpretation of the SQLCA information.

Figure 6 on page 40 shows an example of a receive buffer containing an error SQLCA, and the formatted display of the SQLCA. This SQLCA is the result of an attempt to delete rows from a nonexistent table.

```
1       DB2 fnc_data      gateway_drda_ar      sqljcrecv (1.35.10.81)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 178

        RECEIVE BUFFER:  SQLCARD OBJDSS     (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0065D0030001005F 240800FFFFFF34F4   .e....._$.....4.  ..}....¬.......4
  0010  F2F7F0F4C4E2D5E7 D6E3D34000E2C1D5   ...........@....  2704DSNXOTL .SAN
  0020  6DC6D9C1D5C3C9E2 C3D64040404040FF   m.........@@@@@.  _FRANCISCO     .
  0030  FFFE0C0000000000 000000FFFFFFFF00   ................  ................
  0040  0000000000000040 4040404040404040   .......@@@@@@@@@  .......
  0050  40400000000FC4C4 C3E2E4E2F14BD4E8   @@...........K..    ....DDCSUS1.MY
  0060  E3C1C2D3C5                          .....             TABLE

2       DB2 fnc_data      gateway_drda_ar      sqljmsca (1.35.10.108)
        pid 48732; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 179
        SQLCA

        SQLCAID:  SQLCA
        SQLCABC:  136
        SQLCODE:  -204
        SQLERRML: 15
        SQLERRMC: DDCSUS1.MYTABLE
        SQLERRP:  DSNXOTL
        SQLERRDﬄ0->5": FFFFFE0C, 00000000, 00000000, FFFFFFFF, 00000000, 00000000
        SQLWARN(0->A):  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,
        SQLSTATE: 42704
```

Figure 6. Receive Buffer Example

Chapter 6. Security

This chapter describes DB2 Connect security considerations including Authentication types. It also provides some additional hints and tips on security for DB2 for OS/390 users.

If you wish to know more about setting up security with DCE please refer to the DB2 Universal Database *Administration Guide*, and the database and DCE manuals for your DRDA server platform.

Note: When using DB2 Connect with DCE security, DCE software is required to be installed at the DB2 client workstation, and on the DRDA server, but it is not necessary to install it on the DB2 Connect workstation. For further information about software prerequisites for DCE, please refer to your DB2 Connect *Quick Beginnings* manual.

Authentication

As DB2 Connect administrator, in cooperation with your DRDA database administrator, you can determine where user names and passwords are validated. There are five possibilities:

- Validation at the client
- Validation at the DB2 Connect workstation
- Validation at both the DB2 Connect workstation and the DRDA server
- Validation at the DRDA server
- Validation at a DCE security server.

You determine where validation occurs by setting the Authentication type parameter in the system database directory, and the Security type parameter in the node directory for APPC nodes. For more information about updating these directories, see Chapter 3, "Updating Database Directories" on page 13.

Notes:

1. DB2 Connect itself performs no user validation. If you want to have the DB2 Connect workstation perform validation, the local security subsystem will be used to verify the userid and password provided with each `CONNECT` request. Therefore, when you set up a DB2 Connect Enterprise Edition gateway, if you will use `AUTHENTICATION=SERVER`, you must set up all the necessary userids and passwords on the gateway system.
2. If you use DCE Directory Services, authentication works differently. For more information, see "Security with DCE Directory Services" on page 97.

The following authentication types are allowed with DB2 Connect Version 5:

CLIENT The user name and password are validated at the client.

SERVER The user name and password are validated at the DB2 Connect workstation. When no authentication is specified, **SERVER** is assumed.

DCS The user name and password are validated at the DRDA server.
DCE The user name and password are validated at the DCE security server.

Notes:

1. For any system database directory entry that DB2 Connect uses for establishing a connection, if the authentication parameter is not specified, then DB2 Connect will use authentication **SERVER**.
2. As with DB2 Universal Database client-server communications, the authentication type is not required at a remote client attached to a DB2 Connect Enterprise Edition gateway, but it may be specified there in order to help optimize performance, since then it does not need to be gotten from the gateway, thus reducing the elapsed time for transactions.
3. In the case of a discrepancy between the value at the client and value at the gateway, the value specified at the DB2 Connect gateway takes precedence.

Security Types

This section lists the various combinations of authentication and security settings that are supported with DB2 Connect Version 5.0 over both APPC and TCP/IP connections.

The discussion which follows applies to both types of connection.

Security Types for APPC Connections

The following security types are allowed for APPC connections, in order to specify what security information will flow at the communications layer:

SAME Only the user name is passed to the DRDA server.
PROGRAM The user name and password are passed to the DRDA server.
NONE No security information flows.

Table 2 shows the possible combinations of these values and the authentication type specified on the DB2 Connect workstation, and where validation is performed for each combination. Only the combinations shown in this table are supported by DB2 Connect over APPC connections.

Table 2. Valid Security Scenarios for APPC connections

Case	DB2 Connect Authentication	Security	Validation
1	CLIENT	SAME	Client
2	SERVER	SAME	DB2 Connect workstation
3	SERVER	PROGRAM	DB2 Connect workstation and DRDA server
4	DCS	PROGRAM	DRDA server
5	DCE	NONE	DCE security server

If remote clients are connected to a DB2 Connect Enterprise Edition gateway, specify the following:

- If a remote client is connected to a DB2 Connect gateway via APPC, specify a security type of NONE at the remote client.
- If the authentication type in the database manager configuration at the DB2 Connect gateway is CLIENT, specify CLIENT at each remote client.
- If the authentication type at the DB2 Connect gateway is either SERVER or DCS, specify either SERVER or DCS at each remote client. (Which of these two values you specify at the remote client makes no difference.)

Notes:

1. For AIX systems, all users using APPC security type SAME must belong to the AIX **system** group.
2. For AIX systems with remote clients, the instance of the DB2 Connect product running on the DB2 Connect workstation must belong to the AIX **system** group.
3. Access to a DRDA server is controlled by its own security mechanisms or subsystems; for example, the Virtual Telecommunications Access Method (VTAM) and Resource Access Control Facility (RACF). Access to protected database objects is controlled by the SQL **GRANT** and **REVOKE** statements.

Security Types for TCP/IP Connections

The TCP/IP communication protocol does not support security options at the network protocol layer. Thus only the authentication type controls where authentication takes place. Only the combinations shown in this table are supported by DB2 Connect over TCP/IP connections.

Table 3. Valid Security Scenarios for TCP/IP connections

Case	DB2 Connect Workstation Authentication type	Validation
1	CLIENT	Client
2	SERVER	DB2 Connect workstation
3	Not applicable	None
4	DCS	DRDA server
5	DCE	DCE security server

Discussion of Security Types

The following discussion applies to both APPC and TCP/IP connections, as described above and listed in Table 2 on page 42 and Table 3. Each case is described in more detail, as follows:

- In case 1, the user name and password are validated only at the remote client. (For a local client, the user name and password are validated only at the DB2 Connect workstation.)

  The user is expected to be authenticated at the location he or she first signs on to. The user ID is sent across the network, but not the password. Use this type of security only if all client workstations have adequate security facilities that can be trusted.

- In case 2, the user name and password are validated at the DB2 Connect workstation only. The password is sent across the network from the remote client to the DB2 Connect workstation but not to the DRDA server.
- In case 3, the user name and password are validated at both the DB2 Connect workstation and the DRDA server. The password is sent across the network from the remote client to the DB2 Connect workstation and from the DB2 Connect workstation to the DRDA server.

  Because validation is performed in two places, the same set of user names and passwords must be maintained at both the DB2 Connect workstation and the DRDA server.

- In case 4, the user name and password are validated at the DRDA server only. The user ID and password are sent across the network from the remote client to the DB2 Connect workstation and from the DB2 Connect workstation to the DRDA server.
- In case 5, a DCE encrypted ticket is obtained by the client from the DCE security server. The ticket is passed unaltered through DB2 Connect to the server, where it is validated by the server using DCE Security Services.

Additional Hints and Tips About Security

This section provides some additional hints and tips about security for users of DB2 Connect Version 5.0.

Extended Security Codes

Until DB2 for OS/390 Version 5.1, connect requests that provided user IDs or passwords could fail with SQL30082 reason code 0, but no other indication as to what might be wrong.

DB2 for OS/390 Version 5.1 introduces an enhancement which provides support for extended security codes. Specifying extended security will provide additional diagnostics, such as (PASSWORD EXPIRED) in addition to the reason code.

In order to exploit this, the DB2 for OS/390 ZPARM installation parameter for extended security should be set to the value YES. Use the DB2 for OS/390 installation panel DSN6SYSP to set EXTSEC=YES. You can also use DDF panel 1 (DSNTIPR) to set this. The default value is EXTSEC=NO.

TCP/IP Security Already Verified

If you wish to provide support for the DB2 Universal Database security option `AUTHENTICATION=CLIENT`, then use DB2 for OS/390 installation panel DSNTIP4 (DDF panel 2) to set TCP/IP already verified security to `YES`.

ODBC Application Security

ODBC applications use dynamic SQL. This may create security concerns in some installations. DB2 for OS/390 introduces a new bind option `DYNAMICRULES(BIND)` that allows execution of dynamic SQL under the authorization of either the owner or the binder.

DB2 Universal Database Version 5.0 provides a new CLI/ODBC configuration parameter `CURRENTPACKAGESET` in the DB2CLI.INI configuration file. This should be set to a schema name that has the appropriate privileges. An SQL `SET CURRENT PACKAGESET schema` statement will automatically be issued after every connect for the application.

Use the ODBC Manager to update DB2CLI.INI. See *Installing and Configuring DB2 Clients* for further information.

Chapter 7. Programming in a DRDA Environment

This section provides some information about creating applications that use DB2 Connect. For more information, see the *Road Map to DB2 Programming* and the *Command Reference*.

Programming in a Distributed Environment

DB2 Connect lets an application program access data in any DRDA server database. For example, an application running on OS/2 can access data in a DB2 for OS/390 database. You can create new applications, or modify existing applications to run in a DRDA environment. You can also develop applications in one environment and port them to another.

DB2 Connect enables you to use the following items with host database products such as DB2 for OS/390, as long as the item is supported by the host database product:

- Embedded SQL, both static and dynamic
- Compound SQL, as described in “NOT ATOMIC Compound SQL” on page 55
- Stored procedures, as described in “Stored Procedures” on page 54
- The DB2 Call Level Interface
- The Microsoft ODBC API
- JDBC.

Some SQL statements differ among relational database products. You may encounter the following situations:

- SQL statements that are the same for all the database products that you use regardless of standards
- SQL statements that are documented in the *SQL Reference* and are therefore available in all IBM relational database products
- SQL statements that are unique to one database system that you access.

SQL statements in the first two categories are highly portable, but those in the third category will first require changes.

In general, SQL statements in Data Definition Language (DDL) are not as portable as those in Data Manipulation Language (DML).

DB2 Connect accepts some SQL statements that are not supported by DB2 Universal Database. DB2 Connect passes these statements on to the DRDA server.

For information on limits on different platforms, such as the maximum column length, see the sections on *SQL Limits* and *Considerations for Using* in the *SQL Reference*.

If you move a CICS application from OS/390 or VSE to run under another CICS product (for example, CICS for AIX), it can also access the OS/390 or VSE database using DB2 Connect. Refer to the *CICS/6000 Application Programming Guide* and the *CICS Customization and Operation* manual for more details.

Note: You can use DB2 Connect with a DB2 Universal Database Version 5 database using DRDA, although it would be more efficient to use the DB2 private protocol without DB2 Connect. Most of the incompatibility issues listed in the following sections will not apply if you are using DB2 Connect against a DB2 Universal Database Version 5 database, except in cases where a restriction is due to a limitation of DRDA itself, for example, the non-support of large object (LOB) data types.

When you program in a DRDA environment, you should consider the following specific factors:

- Using Data Definition Language (DDL)
- Using Data Manipulation Language (DML)
- Using Data Control Language (DCL)
- Connecting and disconnecting
- Precompiling
- Defining a sort order
- Managing referential integrity
- Locking
- Differences in SQLCODEs and SQLSTATEs
- Using system catalogs
- Isolation levels
- Stored procedures
- NOT ATOMIC compound SQL
- Distributed unit of work
- SQL statements supported or rejected by DB2 Connect.

Using Data Definition Language (DDL)

DDL statements differ among the IBM database products because storage is handled differently on different systems. On DRDA server systems, there can be several steps between designing a database and issuing a CREATE TABLE statement. For example, a series of statements may translate the design of logical objects into the physical representation of those objects in storage.

The precompiler passes many such DDL statements to the DRDA server when you precompile to a DRDA server database. The same statements would not precompile against a database on the system where the application is running. For example, in an OS/2 application the CREATE STORGROUP statement will precompile successfully to a DB2 for OS/390 database, but not to a DB2 for OS/2 database.

Using Data Manipulation Language (DML)

In general, DML statements are highly portable. SELECT, INSERT, UPDATE, and DELETE statements are similar across the IBM relational database products. Most applications primarily use DML SQL statements, which are supported by the DB2 Connect program and DRDA protocol.

Numeric Data Types

When numeric data is transferred to DB2 Universal Database, the data type may change. Numeric and zoned decimal SQLTYPEs (supported by DB2 for AS/400) are converted to fixed (packed) decimal SQLTYPEs.

Mixed-Byte Data

Mixed-byte data can consist of characters from an extended UNIX code (EUC) character set, a double-byte character set (DBCS) and a single-byte character set (SBCS) in the same column. On systems that store data in EBCDIC (OS/390, OS/400, VSE, and VM), shift-out and shift-in characters mark the start and end of double-byte data. On systems that store data in ASCII (such as OS/2 and UNIX), shift-in and shift-out characters are not required.

If your application transfers mixed-byte data from an ASCII system to an EBCDIC system, be sure to allow enough room for the shift characters. For each switch from SBCS to DBCS data, add 2 bytes to your data length. For better portability, use variable-length strings in applications that use mixed-byte data.

Long Fields

Long fields (strings longer than 254 characters) are handled differently on different systems. A DRDA server may support only a subset of scalar functions for long fields; for example, DB2 for MVS/ESA allows only the **LENGTH** and **SUBSTR** functions for long fields. Also, a DRDA server may require different handling for certain SQL statements; for example, DB2 for VSE & VM requires that with the INSERT statement, only a host variable, the SQLDA, or a NULL value be used.

Large Object (LOB) Data Type

The LOB data type is not supported by the DRDA architecture. DB2 Connect cannot send data with this data type.

Using Data Control Language (DCL)

Each IBM relational database management system provides different levels of granularity for the GRANT and REVOKE SQL statements. Check the product-specific publications to verify the appropriate SQL statements to use for each database management system.

Connecting and Disconnecting

DB2 Connect supports the CONNECT TO and CONNECT RESET versions of the CONNECT statement, as well as CONNECT with no parameters. If an application calls an SQL statement without first performing an explicit CONNECT TO statement, an *implicit* connect is performed to the default application server (if one is defined).

When you connect to a database, information identifying the relational database management system is returned in the SQLERRP field of the SQLCA. If the application server is an IBM relational database, the first three bytes of SQLERRP contain one of the following:

DSN DB2 for MVS/ESA and DB2 for OS/390

ARI	DB2 for VSE & VM
QSQ	DB2 for AS/400
SQL	DB2 Universal Database.

If you issue a CONNECT TO or null CONNECT statement while using DB2 Connect, the country code or territory token in the SQLERRMC field of the SQLCA is returned as blanks; the CCSID of the application server is returned in the code page or code set token.

You can explicitly disconnect by using the CONNECT RESET statement (for type 1 connect), the RELEASE and COMMIT statements (for type 2 connect), or the DISCONNECT statement (either type of connect, but not in a TP monitor environment). Type 2 connect is used for distributed unit of work, as described in “Distributed Unit of Work” on page 10.

If a connection is not explicitly disconnected and the application ends normally, DB2 Connect commits the resulting data implicitly.

Note: An application can receive SQLCODEs indicating errors and still end normally; DB2 Connect commits the data in this case. If you do not want the data to be committed, you must issue a ROLLBACK command.

The FORCE command lets you disconnect selected users or all users from the database. This is supported for DRDA server databases; the user can be forced off the DB2 Connect workstation.

Precompiling

There are some differences in the precompilers for different IBM relational database systems. The precompiler for DB2 Universal Database differs from the DRDA server precompilers in the following ways:

- It makes only one pass through an application.
- It does not support the DB2 for MVS/ESA DCLGEN command, which is frequently used in application development on MVS.
- When binding against DB2 Universal Database databases, objects must exist for a successful bind. VALIDATE RUN is not supported.

Blocking

The DB2 Connect program supports the DB2 database manager blocking bind options:

UNAMBIG	Only unambiguous cursors are blocked (the default).
ALL	Ambiguous cursors are blocked.
NO	Cursors are not blocked.

The DB2 Connect program uses the block size defined in the DB2 database manager configuration file for the RQRIOBLK field. Current versions of DRDA supports block sizes up to 32 767. If larger values are specified in the DB2 database manager configuration file, DB2 Connect uses a value of 32 767 but does not reset the DB2 database manager configuration file. Blocking is handled the same way using the same block size for dynamic and static SQL.

Note: Most DRDA server systems consider dynamic cursors ambiguous, but DB2 Universal Database systems consider some dynamic cursors unambiguous. To avoid confusion, you can specify BLOCKING ALL with DB2 Connect.

Specify the block size in the DB2 database manager configuration file by using the CLP, the Control Center, or an API, as listed in the *API Reference* and *Command Reference*.

Package Attributes

A package has the following attributes:

Collection ID The ID of the package. It can be specified on the PREP command.

Owner The authorization ID of the package owner. It can be specified on the PREP or BIND command.

Creator The user name that binds the package.

Qualifier The implicit qualifier for objects in the package. It can be specified on the PREP or BIND command.

Each DRDA server system has limitations on the use of these attributes:

DB2 for MVS/ESA and DB2 for OS/390 All four attributes can be different. The use of a different qualifier requires special administrative privileges.

DB2 for VSE & VM All of the attributes must be identical. If USER1 creates a bind file (with PREP), and USER2 performs the actual bind, USER2 needs DBA authority to bind for USER1. Only USER1's user name is used for the attributes.

DB2 for AS/400 The qualifier indicates the collection name. The relationship between qualifiers and ownership affects the granting and revoking of privileges on the object. The user name that is logged on is the creator and owner unless it is qualified by a collection ID, in which case the collection ID is the owner. The collection ID must already exist before it is used as a qualifier.

DB2 Universal Database For DB2 Universal Database databases, the collection ID and the creator can be different. The creator also becomes the owner and the qualifier of the package.

Note: DB2 Connect provides support for the *SET CURRENT PACKAGESET* command for DB2 for MVS/ESA, DB2 for OS/390, and DB2 Universal Database.

C Null-terminated Strings

The CNULREQD bind option overrides the handling of null-terminated strings that are specified using the LANGLEVEL option. Refer to *Embedded SQL Programming Guide* for a description of how null-terminated strings are handled when prepared with the LANGLEVEL option set to MIA or SAA1. By default, CNULREQD is set to YES. This causes null-terminated strings to be interpreted according to MIA standards. If connecting to a DB2 for OS/390 server it is strongly recommended to set CNULREQD to YES. DB2 Connect will use CNULREQD as a default. You need to bind applications coded to SAA1 standards (with respect to null-terminated strings) with the CNULREQD option set to NO. Otherwise, null-terminated strings will be interpreted according to MIA standards, even if they are prepared using LANGLEVEL set to SAA1.

Standalone SQLCODE and SQLSTATE

Standalone SQLCODE and SQLSTATE variables, as defined in ISO/ANS SQL92, are supported through the LANGLEVEL SQL92E precompile option. An SQL0020W warning will be issued at precompile time, indicating that LANGLEVEL is not supported. This warning applies only to the features listed under LANGLEVEL MIA in the *Command Reference*, which is a subset of LANGLEVEL SQL92E.

Defining a Sort Order

The differences between EBCDIC and ASCII cause differences in sort orders in the various database products, and also affect ORDER BY and GROUP BY clauses. One way to minimize these differences is to create a user-defined collating sequence that mimics the EBCDIC sort order. You can specify a collating sequence only when you create a new database. For more information, see the *Embedded SQL Programming Guide*, the *API Reference* and the *Command Reference*.

Note: Database tables can now be stored on DB2 for OS/390 in ASCII format. This permits faster exchange of data between DB2 Connect and DB2 for OS/390, and removes the need to provide field procedures which must otherwise be used to convert data and resequence it.

Managing Referential Integrity

Different systems handle referential constraints differently:

DB2 for MVS/ESA or DB2 for OS/390
An index must be created on a primary key before a foreign key can be created using the primary key. Tables can reference themselves.

DB2 for VSE & VM
An index is automatically created for a foreign key. Tables cannot reference themselves.

DB2 for AS/400
An index is automatically created for a foreign key. Tables can reference themselves.

DB2 Universal Database For DB2 Universal Database databases, an index is automatically created for a foreign key. Tables can reference themselves.

Other rules vary concerning levels of cascade.

Locking

The way in which the database server performs locking can affect some applications. For example, applications designed around row-level locking and the isolation level of cursor stability are not directly portable to systems that perform page-level locking. Because of these underlying differences, applications may need to be adjusted.

The DB2 for OS/390 and DB2 Universal Database products have the ability to time-out a lock and send an error return code to waiting applications.

Differences in SQLCODEs and SQLSTATEs

Different IBM relational database products do not always produce the same SQLCODEs for similar errors. You can handle this problem in either of two ways:

- Use the SQLSTATE instead of the SQLCODE for a particular error.

  SQLSTATEs have approximately the same meaning across the database products, and the products produce SQLSTATEs that correspond to the SQLCODEs.

- Map the SQLCODEs from one system to another system.

  By default, DB2 Connect maps SQLCODEs and tokens from each IBM DRDA server system to your DB2 Universal Database system. You can specify your own SQLCODE mapping file if you want to override the default mapping or you are using a DRDA server that does not have SQLCODE mapping (a non-IBM DRDA server). You can also turn off SQLCODE mapping. For more information, see Chapter 8, “SQLCODE Mapping” on page 61.

Using System Catalogs

The system catalogs vary across the IBM database products. Many differences can be masked by the use of views. For information, see the documentation for the database server that you are using.

The catalog functions in CLI get around this problem by presenting support of the same API and result sets for catalog queries across the DB2 family (including DRDA).

Numeric Conversion Overflows on Retrieval Assignments

Numeric conversion overflows on retrieval assignments may be handled differently by different IBM relational database products. For example, consider fetching a float column into an integer host variable from DB2 for OS/390 and from DB2 Universal Database. When converting the float value to an integer value, a conversion overflow may occur. By default, DB2 for OS/390 will return a warning sqlcode and a null value to the application. In contrast, DB2 Universal Database will return a conversion overflow error. It is recommended that applications avoid numeric conversion overflows on retrieval assignments by fetching into appropriately sized host variables.

Isolation Levels

DB2 Connect accepts the following isolation levels when you prep or bind an application:

RR Repeatable Read
RS Read Stability
CS Cursor Stability
UR Uncommitted Read
NC No Commit

The isolation levels are listed in order from most protection to least protection. If the DRDA server does not support the isolation level that you specify, the next higher supported level is used.

Table 4 shows the result of each isolation level on each DRDA application server.

Table 4. Isolation Levels

DB2 Connect	DB2 for MVS/ESA or DB2 for OS/390	DB2 for VSE & VM	DB2 for AS/400	DB2 Universal Database
RR	RR	RR	note 1	RR
RS	note 2	RR	COMMIT(*ALL)	RS
CS	CS	CS	COMMIT(*CS)	CS
UR	note 3	CS	COMMIT(*CHG)	UR
NC	note 4	note 5	COMMIT(*NONE)	UR

Notes:

1. There is no equivalent COMMIT option on DB2 for AS/400 that matches RR. DB2 for AS/400 supports RR by locking the whole table.
2. Results in RR for Version 3.1, and results in RS for Version 4.1 with APAR PN75407 or Version 5.1.
3. Results in CS for Version 3.1, and results in UR for Version 4.1 or Version 5.1.
4. Results in CS for Version 3.1, and results in UR for Version 4.1 with APAR PN60988 or Version 5.1.
5. Isolation level NC is not supported with DB2 for VSE & VM.

With DB2 for AS/400, you can access an unjournalled table if an application is bound with an isolation level of UR and blocking set to ALL, or if the isolation level is set to NC.

Stored Procedures

- Invocation

A client program can invoke a server program by issuing an SQL CALL statement. Each server works a little differently to the other servers in this case.

MVS or OS/390 The procedure name must not be more than 8 bytes long and must be defined in the SYSIBM.SYSPROCEDURES catalog on the server.

VSE or VM Stored procedures are not supported.

OS/400 The procedure name must be an SQL identifier. You can also use the DECLARE PROCEDURE or CREATE PROCEDURE statements to specify the actual path name (the schema-name or collection-name) to locate the stored procedure.

All CALL statements to DB2 for AS/400 from REXX/SQL must be dynamically prepared and executed by the application as the CALL statement implemented in REXX/SQL maps to CALL USING DESCRIPTOR.

For the syntax of the SQL CALL statement, see the *SQL Reference*. For information on how to use stored procedures when writing application programs, see the *Embedded SQL Programming Guide*.

The server program on DB2 Universal Database is invoked with a different parameter convention than a server program on DB2 for MVS/ESA, DB2 for OS/390, or DB2 for AS/400. For more information on the parameter convention for a specific platform, refer to the DB2 product documentation for that platform.

All the SQL statements in a stored procedure are executed as part of the SQL unit of work started by the client SQL program.

- Do not pass indicator values with special meaning to or from stored procedures.

  Between DB2 Universal Database, the systems pass whatever you put into the indicator variables. However, when using the DRDA protocol (such as through DB2 Connect), you can only pass 0, -1, and -128 in the indicator variables.

- You should define a parameter to return any errors or warning encountered by the server application.

  A server program on DB2 Universal Database can update the SQLCA to return any error or warning, but a stored procedure on DB2 for OS/390, DB2 for AS/400, or DB2 for MVS/ESA has no such support. If you want to return an error code from your stored procedure, you must pass it as a parameter. The SQLCODE and SQLCA is only set by the server for system detected errors.

NOT ATOMIC Compound SQL

Compound SQL allows multiple SQL statements to be grouped into a single executable block. This may reduce network overhead and improve response time.

DB2 Connect supports NOT ATOMIC compound SQL. This means that processing of compound SQL continues following an error. (With ATOMIC compound SQL, which is not supported by DB2 Connect, an error would roll back the entire group of compound SQL.)

Statements will continue execution until terminated by the application server. In general, execution of the compound SQL statement will be stopped only in the case of serious errors.

NOT ATOMIC compound SQL can be used with all of the supported DRDA application servers.

If multiple SQL errors occur, the SQLSTATEs of the first seven failing statements are returned in the SQLERRMC field of the SQLCA with a message that multiple errors occurred. For more information, see the *SQL Reference*.

Distributed Unit of Work with DB2 Connect

DB2 Connect allows you to update multiple databases within a single distributed unit of work (DUOW). Whether you can use this capability depends on several factors:

- Your application program must be precompiled with the CONNECT 2 and SYNCPOINT TWOPHASE options.
- Databases which are connected over SNA links (node type APPC) can be updated using two-phase commit (2PC) through DB2 Connect in a single unit of work, *as long as:*
  - The LU 6.2 Syncpoint Manager is installed on AIX or OS/2.
  - The right software prerequisites are installed and configured appropriately: IBM SNA Server for AIX Version 3.1 or higher, or IBM Communications Server for OS/2 Version 4 or higher. (For further information refer to the chapter *Setting Up Two-phase Commit Using SNA* in *DB2 Connect Enterprise Edition Quick Beginnings*).
  - The databases are one of:
    - DB2 for OS/390, Version 5.1 or later
    - DB2 for AS/400, Version 3.1 or later
    - DB2 for MVS/ESA, Version 3.1 or later
    - DB2 for VSE & VM, Version 5.1 or later

    This is regardless of whether CICS or Encina is being used.

    Additionally, the DB2 Connect Enterprise Edition Version 5 Syncpoint Manager continues to allow DB2 MVS Version 3.1 databases to participate in two-phase commit, but *not* to act as the *TM_DATABASE* for such transactions.
- DB2 for OS/390 Version 5 databases connected over TCP/IP links (node type TCPIP) can also participate in units of work using two-phase commit, and they are also able to participate with DB2 Universal Database Version 5 databases in the same transaction. When using TCP/IP, DB2 for OS/390 can be defined as the TM_DATABASE.

Notes:

1. DB2 Common Server Version 2.1 databases can be updated with two-phase commit in a unit of work *only* when DB2 for OS/390 Version 5.1 is *not* the transaction manager database.
2. If the TM_DATABASE for the transaction is DB2 for OS/390 Version 5.1, then DB2 CS V2.1 databases participating in that transaction become *READ-ONLY* for the client application.
3. Mixed connection scenarios for DB2 Universal Database Version 5 client applications are only supported if DB2 for OS/390 Version 5.1 is the *TM_DATABASE* for the transaction. For further information refer to the chapter *Setting Up Two-phase Commit Using TCP/IP* in *DB2 Connect Enterprise Edition Quick Beginnings*.
4. The database levels supported for two-phase commit transactions over TCP/IP depend on the Client Application Enabler level, the TM_DATABASE level, and the participant database levels.

   We strongly recommend that *all* clients accessing DB2 Universal Database Version 5 and DB2 for OS/390 Version 5 databases be at DB2 Universal Database Version 5 level. DB2 Version 2.1 clients *cannot* initiate two-phase commit transactions if DB2 for OS/390 Version 5 databases participate in the transaction.

DRDA Server SQL Statements Supported by DB2 Connect

The following statements compile successfully for DRDA server processing but not for processing with DB2 Universal Database systems:

- ACQUIRE
- DECLARE (modifier.(qualifier.)table_name TABLE ...
- LABEL ON

These statements are also supported by the command line processor.

The following statements are supported for DRDA server processing but are not added to the bind file or the package and are not supported by the command line processor:

- DESCRIBE statement_name INTO descriptor_name USING NAMES
- PREPARE statement_name INTO descriptor_name USING NAMES FROM ...

The precompiler makes the following assumptions:

- Host variables are input variables
- The statement is assigned a unique section number.

DRDA Server SQL Statements Rejected by DB2 Connect

The following SQL statements are not supported by DB2 Connect and not supported by the command line processor:

- COMMIT WORK RELEASE
- DECLARE state_name, statement_name STATEMENT

- DESCRIBE statement_name INTO descriptor_name USING xxxx (where xxxx is ANY, BOTH, or LABELS)
- PREPARE statement_name INTO descriptor_name USING xxxx FROM :host_variable (where xxxx is ANY, BOTH, or LABELS)
- PUT ...
- ROLLBACK WORK RELEASE
- SET :host_variable = CURRENT ...

DB2 for VSE & VM extended dynamic SQL statements are rejected with -104 and syntax error SQLCODEs.

Implementing Chargeback Accounting

Many DB2 for OS/390 installations implement resource monitoring practices which allow systems administrators to associate resource usage with individual user access. This can be used to charge individual users or their departments for the resources they consume. This practice is commonly referred to as *charge-back accounting*. DB2 Connect products allow system administrators to monitor mainframe resources consumed by users accessing databases through DB2 Connect.

You can use accounting strings to send accounting data from DB2 Connect to the DRDA application server. An accounting string combines system-generated data with user-supplied data. This data lets system administrators associate resource usage with each user's access, and charge the users accordingly.

The accounting string is sent using the DRDA parameter PRDDTA. Because the content of this parameter is not architected in DRDA, there is no guarantee that your application server will recognize the data as accounting data. Currently, PRDDTA is supported as follows:

MVS and OS/390	The string is stored as an accounting record.
VSE or VM	The string is ignored.
OS/400	The string is ignored.
other	For OS/2 and UNIX-based systems, the string is ignored.

The accounting string consists of 56 bytes generated by DB2 Connect (the prefix) followed by up to 199 bytes specified by the user (the suffix), for a maximum length of 255.

Table 5 on page 59 shows the system-generated fields. Each of these fields is padded to the right with blanks.

Table 5. Accounting String Fields Generated by DB2 Connect

Field Name	Length	Description.
acct_str_len	1	A hexadecimal value representing the length of the accounting string minus 1. For example, X'3C'.
client_prdid	8	The product ID of the client's Client Application Enabler software. For example, the product ID for DB2 Universal Database Version 5.0 is SQL05000.
client_platform	18	The platform the client is on, for example AIX, OS/2, DOS, or Windows.
client_appl_name	20	The first 20 characters of the user's application name, for example, payroll.
client_authid	8	The authid of the user's application, for example, SMITH.
suffix_len	1	A hexadecimal value representing the length of the user-supplied suffix. X'00' means that there is no user-supplied suffix.

The user-defined suffix is one of the following:

- The value specified by an application with the sqlesact() API
- The value of the DB2ACCOUNT environment variable
- The value of the DFT_ACCOUNT_STR (default accounting string) configuration parameter
- A null string.

If the suffix is longer than 199 characters, it is truncated. To be sure that the accounting string is converted correctly when it is transmitted to the DRDA server, you should use only the characters A to Z, 0 to 9 and underscore (_).

The API method of setting the accounting string is recommended. Your application should call the API before connecting to a database. If you want to change the accounting string within the application (for example, to send a different string when you connect to a different database), call the API again. Otherwise, the value remains in effect until the end of the application.

If the sqlesact() API is not called before the first database connection request, the DB2ACCOUNT environment variable is read. This value remains in effect until the end of the application or background command line processor process. To specify a new accounting string suffix after the first database connection, either use the sqlesact() API or end the application or background CLP process and restart it with DB2ACCOUNT set to its new value.

If no DB2ACCOUNT value exists, the value of the DFT_ACCOUNT_STR system configuration parameter is used. This default can be useful for database clients that do not have the ability to forward an accounting string to DB2 Connect. If this does not exist, a null string is used.

The following are examples of accounting strings:

```
x'3C'SQL02010OS/2            cheque            SMITH   x'05'DEPT1

x'37'SQL02010OS/2            cheque            SMITH   x'00'
```

In the first example, the user-defined suffix is DEPT1. In the second example, it is a null string.

Useful Publications

The following publications may help you develop applications that run in a distributed environment:

- The application programming books for the specific database products may contain information that differs from one product to another.
- The SQL reference books for the specific database products will help you ensure that an application contains only supported SQL statements, with the correct syntax.
- The *DB2 for OS/390 Version 5 Reference for Remote DRDA Requesters and Servers* provides the latest information about chargeback accounting for DB2 for OS/390 users.
- *IBM SQL Reference* provides a high-level discussion of differences among the IBM relational database products, and also discusses how to handle some specific differences.
- The DRDA publications provide information on planning, connectivity, programming, and problem determination in the DRDA environment. For a list of titles and order numbers, see “Related DRDA Publications” on page viii.

Chapter 8. SQLCODE Mapping

Different IBM relational database products do not always produce the same SQLCODEs for similar errors. Even when the SQLCODE is the same, it may be accompanied by *tokens* that are specified differently. (The token list is passed in the SQLERRMC field of the SQLCA.) By default, DB2 Connect maps SQLCODEs and tokens from each IBM DRDA server system to your OS/2 or UNIX-based system.

Turning Off SQLCODE Mapping

If you want to turn off SQLCODE mapping, specify **NOMAP** in the parameter string of the DCS Directory or the DCE routing information object. For information about updating the DCS directory, see Chapter 3, “Updating Database Directories” on page 13. For information about using DCE, see Appendix C, “Using DCE Directory Services” on page 91.

If you port an application directly from a DRDA server system (such as DB2 for OS/390), you might want to turn off SQLCODE mapping. This would let you use the application without changing the SQLCODEs that it references.

Tailoring the SQLCODE Mapping

By default, DB2 Connect maps SQLCODEs and tokens from each IBM DRDA server system to your OS/2 or UNIX-based system. The following files are copies of the default SQLCODE mapping:

dcs1dsn.map	Maps DB2 for MVS/ESA and DB2 for OS/390 SQLCODEs
dcs1ari.map	Maps DB2 for VSE & VM SQLCODEs
dcs1qsq.map	Maps DB2 for AS/400 SQLCODEs

No mapping is required for OS/2 and UNIX-based DB2 systems.

If you want to override the default SQLCODE mapping or you are using a DRDA server that does not have SQLCODE mapping (a non-IBM DRDA server), you can copy one of these files and use it as the basis for your new SQLCODE mapping file. (By copying the file rather than editing it directly, you ensure that you can always refer to the original SQLCODE mapping if necessary.)

Specify the file name of your new SQLCODE mapping file in the parameter string of the DCS Directory or the DCE routing information object. For information about updating the DCS directory, see Chapter 3, “Updating Database Directories” on page 13. For information about using DCE, see Appendix C, “Using DCE Directory Services” on page 91.

Each mapping file is an ASCII file, which is created and edited using an ASCII editor. At initial installation, the file is stored in the `map` directory in the installation path.

The file can contain the following special types of lines:

&& The logical beginning of the file. All lines before the first occurrence of && are considered free-form comments and ignored. If the file contains nothing after &&, no SQLCODE mapping is performed. (You can also turn off SQLCODE mapping with the NOMAP parameter, as described previously.)

***** As the first character on a line, indicates a comment.

W As the only character on a line, indicates that warning flags should be remapped. (By default, the original warning flags are passed.) The W must be uppercase.

All other lines after && must be either blank or mapping statements in the following form:

```
input_code [, output_code [, token_list]]
```

input_code is one of the following:

sqlcode The SQLCODE from the DRDA server database.

U All undefined negative SQLCODEs (those not listed in this file) are mapped to the specified *output_code*. If no *output_code* is specified on this line, the original SQLCODE is used. This character must be uppercase.

P All undefined positive SQLCODEs (those not listed in this file) are mapped to the specified *output_code*. If no *output_code* is specified on this line, the original SQLCODE is used. This character must be uppercase.

cc*nn* The SQLSTATE class code from the DRDA server database. *nn* is one of the following:

00	Unqualified successful completion
01	Warning
02	No data
21	Cardinality violation
22	Data exception
23	Constraint violation
24	Invalid cursor state
26	Invalid SQL statement identifier
40	Transaction Rollback
42	Access violation
51	Invalid application state
52	SQL or product limit exceeded
55	Object not in prerequisite state
56	Miscellaneous SQL or Product Error
57	Resource not available or operator intervention
58	System error

The specified *output_code* is used for all SQLCODEs with this class code that are not specified explicitly in the mapping file. If no *output_code* is specified on this line, the original SQLCODE is mapped to itself with no tokens copied over.

The characters **cc** must be lowercase.

If the same *input_code* appears more than once in the mapping file, the first occurrence is used.

output_code is the output SQLCODE. The input code is mapped to the output code, and the input code is pushed onto the SQLERRD array within the SQLCA; it becomes the first entry in this array and the previous entries 1-5 become entries 2-6. If no value is specified, the original SQLCODE is used.

If you specify an output code, you can also specify one of the following:

(s) The input SQLCODE plus the product ID (ARI, DSN or QSQ) will be put into the SQLCA message token field.

The original SQLCODE is returned as the only token. This option is designed to handle undefined SQLCODEs, with the exception of +965 and -969. If +965 or -969 is the *output_code*, the token list returned in the SQLERRMC field of the SQLCA includes the original SQLCODE, followed by the product identifier, followed by the original token list.

The character **s** must be lowercase.

(*token-list***)** A list of tokens, separated by commas. Specify only a comma to skip a particular token. For example, the form (,*t2*,,*t4*) means that the first and third output tokens are null.

Each token has the form of a number (*n*), optionally preceded by **c**, optionally followed by **c** or **i**. It is interpreted as follows:

c The datatype of the token in this position is CHAR (the default). If **c** comes before *n*, it refers to the input token; if it comes after *n*, it refers to the output token. The character **c** must be lowercase.

i The datatype of the token in this position is INTEGER. If **i** comes after *n*, it refers to the output token. **i** should not come before *n*, because IBM DRDA server database products support only CHAR tokens. The character **i** must be lowercase.

n A number or numbers indicating which DRDA server tokens are used. They are arranged in the order desired for placement in the output SQLCA. The number indicates the DRDA server token; the arrangement indicates the order in which the tokens will be placed in the SQLCA.

For example, the DRDA server might return two tokens, 1 and 2. If you want token 2 to appear before token 1 in the output SQLCA, specify (2,1).

Multiple token numbers can be combined to form one CHAR output token by connecting them with periods.

Commas are used to separate output tokens. If no token is specified before a comma, no output token is included in the SQLCA for that position. Any tokens occurring in the output SQLCA following the last specified token are mapped to a null token.

Figure 7 on page 64 shows a sample SQLCODE mapping file.

```
&&
  -007    ,   -007   ,   (1)
  -010
  -060    ,   -171   ,   (2)
...
  -204    ,   -204   ,   (c1.2c)
...
  -633    ,   -206   ,   (,c1i)

  -30021  ,   -30021 ,   (c1c,c2c)

  cc00    ,   +000
...
  U       ,   -969   ,   (s)
  P       ,   +965   ,   (s)
```

Figure 7. An SQLCODE Mapping File

Each mapping statement in the file is described below:

1. The SQLCODE is mapped from -007 to -007. The first input token received from the DRDA server is used as the first output token, and it defaults to CHAR. No other tokens are transferred.
2. The SQLCODE is mapped from -010 to -010 (no output SQLCODE is specified). No tokens are put into the output SQLCA.
3. The SQLCODE is mapped from -060 to -171. The first input token received from the DRDA server is discarded. The second is used as the first token in the output SQLCA, and it is CHAR. There is no second token in the output SQLCA.
4. The SQLCODE is mapped from -204 to -204. The first and second tokens received from the DRDA server are CHAR. These two input tokens are combined to form one CHAR output token, which will be the first output token in the SQLCA.
5. The SQLCODE is mapped from -633 to -206. The first input token received from the DRDA server is CHAR. It is converted to INTEGER and is used as the second token in the output SQLCA. The first token in the output SQLCA is null, as indicated by a comma.
6. The SQLCODE is mapped from -30021 to -30021. The first and second input tokens received from the DRDA server are CHAR, and they are used as the first and second tokens in the output SQLCA.
7. All SQLCODEs in SQLCAs with SQLSTATEs in the 00 class will be mapped to SQLCODE +000.
8. All undefined SQLCODEs are mapped to -969. This option should be used only if all mappable codes are listed, including all those that are identical and require no mapping. The **(s)** option indicates that the token list to be returned in the SQLERRMC field of the SQLCA includes the original SQLCODE, followed by the

product the error occurred in, followed by the original token list. If the **U** entry is not included, all unlisted codes are passed without any mapping.

9. All undefined positive SQLCODEs are mapped to +965. This option should be used only if all mappable codes are listed, including all those that are identical and require no mapping. The **(s)** option indicates that the token list to be returned in the SQLERRMC field of the SQLCA includes the original SQLCODE, followed by the product the warning occurred in, followed by the original token list. If the **P** entry is not included, all unlisted positive codes are passed without any mapping.

Chapter 9. Problem Determination

The DB2 Connect environment involves multiple software, hardware and communications products. Problem determination is best approached by a process of elimination and refinement of the available data to arrive at a conclusion (the location of the error).

The following topics are provided to assist in the problem determination process.

- An initial connection was not successful
- After an initial successful connection problems are encountered
- Diagnostic tools

After gathering the relevant information and based on your selection of the applicable topic proceed to the referenced section.

Other Information Sources

This section lists additional sources of information.

Using the Troubleshooting Guide

For more information on DB2 Connect and DB2 Universal Database problem determination topics, please refer to the *Troubleshooting Guide*.

Using the World Wide Web

You can find the most recent information about DB2 Connect problem determination hints and tips in the DB2 Product and Service Technical Library on the World Wide Web:

1. Set your Web browser to the following URL:

   ```
   http://www.software.ibm.com/data/db2/library/
   ```

2. Select “DB2 Universal Database”.
3. Search for “Technotes” using the keywords “DDCS”, or “Connect”.

Gathering Relevant Information

Problem determination includes narrowing the scope of the problem and investigating the possible causes. The proper starting point is to gather the relevant information and determine what you know, what data has not been gathered and what paths you can eliminate. At a minimum answer the following questions.

- Has the initial connection been successful?
- Is the hardware functioning properly?
- Are the communication paths operational?
- Have there been any communication network changes that would make previous directory entries invalid?

- Has the database been started?
- Is the communication breakdown between client and DB2 Connect workstation, DB2 Connect workstation and DRDA server, all clients or one client?
- What can you determine by the content of the message and the tokens returned in the message?
- Will using diagnostic tools provide any assistance at this time?
- Are other machines performing similar tasks working correctly?
- If this is a remote task, is it successful if performed locally?

Initial Connection is Not Successful

Review the following questions and ensure that the installation steps were followed.

1. *Did the installation processing complete successfully?*
   - Were all the prerequisite software products available?
   - Were the memory and disk space adequate?
   - Was remote client support installed?
   - Was the installation of the communications software completed without any error conditions?
2. *For UNIX-based systems was an instance of the product created?*
   - As root did you create a user and a group to become the instance owner and sysadm group?
3. *If applicable, was the license information processed successfully?*
   - For UNIX-based systems, did you edit the nodelock file and enter the password that IBM supplied?
4. *Were the DRDA server and workstation communications configured properly?*
   - There are three configurations that must be considered:
     a. The DRDA server configuration identifies the application requester to the server. The DRDA server database management system will have system catalog entries that will define the requestor in terms of location, network protocol and security.
     b. The DB2 Connect workstation configuration defines the client population to the server and the DRDA server to the client.
     c. The client workstation configuration must have the name of the workstation and the communications protocol defined.
   - Problem analysis for not making an initial connection includes verifying for SNA connections that all the LU (logical unit) and PU (physical unit) names are complete and correct, or verifying for TCP/IP connections that the correct port number and hostname have been specified.
   - Both the DRDA server database administrator and the Network administrators have utilities available to diagnose problems.

5. *Do you have the level of authority required by the DRDA server database management system to use the DRDA server database?*

   - Consider the access authority of the user, rules for table qualifiers, the anticipated results.

6. *If you attempt to use the command line processor to issue SQL statements against a DRDA server database are you unsuccessful?*

   - Did you follow the procedure to bind the command line processor to the DRDA server database?

Problems Encountered after an Initial Connection

The following questions are offered as a starting point to assist in narrowing the scope of the problem.

1. *Are there any special or unusual operating circumstances?*

   - Is this a new application?
   - Are new procedures being used?
   - Are there recent changes that might be affecting the system? For example, have any of the software products or applications been changed since the application or scenario last ran successfully?
   - For application programs, what application programming interface (API) was used to create the program?
   - Have other applications that use the software or communication APIs been run on the user's system?
   - Has a PTF recently been installed? If the problem occurred when a user tried to use a feature that had not been used (or loaded) on their operating system since it was installed, determine IBM's most recent PTF level and load that level *after* installing the feature.

2. *Has this error occurred before?*

   Is there any documented resolution to previous error conditions?
   Who were the participants and can they provide insight into possible course of action?

3. *Have you explored using communications software commands that return information about the network?*

   Is there a verification tool available for your SNA software?
   If you are using TCP/IP there may be valuable information retrieved from using TCP/IP commands and daemons.

4. *Is there information returned in the SQLCA (SQL communication area) that can be helpful?*

   Problem handling procedures should include steps to examine the contents of the SQLCODE and SQLSTATE fields.

SQLSTATEs allow application programmers to test for classes of errors that are common to the DB2 family of database products. In a distributed relational database network this field may provide a common base. For more information see the *Message Reference*

5. *Was DB2START executed at the Server?* Additionally, ensure that the DB2COMM environment variable is set correctly for clients accessing the server remotely.

6. *Are other machines performing the same task able to connect to the server successfully?* The maximum number of clients attempting to connect to the server may have been reached. If another client disconnects from the server, is the client previously not able to connect, now able to connect?

7. *Does the machine have the proper addressing?* Verify that the machine is unique in the network.

8. *When connecting remotely, has the proper authority been granted to the client?* Connection to the instance may be successful, but authorization not granted at the database or table level.

9. *Is this the first machine to connect to a remote database?* In distributed environments routers or bridges between networks may block communication between the client and the server. For example, when using APPC, ensure that a session can be established. When using TCP/IP, ensure that you can PING the remote host.

Diagnostic Tools

When you encounter a problem, you can use the following:

- The first failure service log, where diagnostic information is consolidated and stored in a readable format. For more information, see the *Troubleshooting Guide*. For information about messages found in the log, see the *Message Reference*.
- `db2diag.log`

  This file is located in `/u/db2/sqllib/db2dump/db2diag.log` on UNIX systems, where `db2` is the instance name.

  This file is located in `x:\sqllib\db2\db2diag.log` on Intel systems, where `x:` is the logical drive, and `db2` is the instance name.
- `db2alert.log` (Same file locations as `db2diag.log`).
- The trace utility, as described in "Trace Utility (ddcstrc)" on page 30.
- For UNIX-based systems, the **ps** command, which returns process status information about active processes to standard output.
- For UNIX-based systems, the core file that is created in the current directory when severe errors occur. It contains a memory image of the terminated process, and can be used to determine what function caused the error.
- For Windows NT systems, use the Event Viewer.

For more information on troubleshooting TCP/IP connections (or other topics), please refer to *Troubleshooting Guide*, or search for “Technotes” on the DB2 Product and Service Technical Library (see “Using the World Wide Web” on page 67.

Chapter 10. DB2 Connect Performance

The DB2 Connect product interacts with many different products including DRDA application server products, client products, and communication products. Its performance depends on all these pieces working together efficiently.

Note: This chapter assumes that all applications are run on the DB2 Connect workstation. If you have remote clients, you must also consider the performance of the remote connections to the DB2 Connect workstation.

Other Information Sources

This section lists additional sources of information.

Other Publications

The following books contain additional information about performance:

- *DDCS for OS/2 to DB2 Performance Benchmark*
- *SNA Server for AIX and SNA Server Gateway for AIX Performance Guide*

Using the World Wide Web

You can find the most recent information about DB2 Connect performance hints and tips in the DB2 Product and Service Technical Library on the World Wide Web:

1. Set your Web browser to the following URL:

   ```
   http://www.software.ibm.com/data/db2/library/
   ```

2. Select “DB2 Universal Database”.
3. Search for “Technotes” using the keywords “DDCS” or “Connect”, and “performance”.

Performance Concepts and Tools

Performance is the way a computer system behaves given a particular workload. It is affected by the resources available and how they are used and shared.

If you want to improve performance, you must first decide what you mean by performance. You can choose many different *performance metrics*, including:

Response time
The interval between the time that the application sends the database request and the time that the application receives a response.

Transaction throughput
The number of units of work that can be completed per unit of time. The unit of work could be simple, like fetching and updating a row, or complicated, involving hundreds of SQL statements.

Data transfer rate

The number of bytes of data transferred between the DB2 Connect application and the DRDA database per unit of time.

Performance will be limited by the available hardware and software resources. CPU, memory and network adapters are examples of hardware resources. Communication subsystems, paging subsystems, `mbuf` for AIX, and `link` for SNA are examples of software resources.

Data Flows

Figure 8 shows the path of data flowing between the DRDA database and the workstation through DB2 Connect.

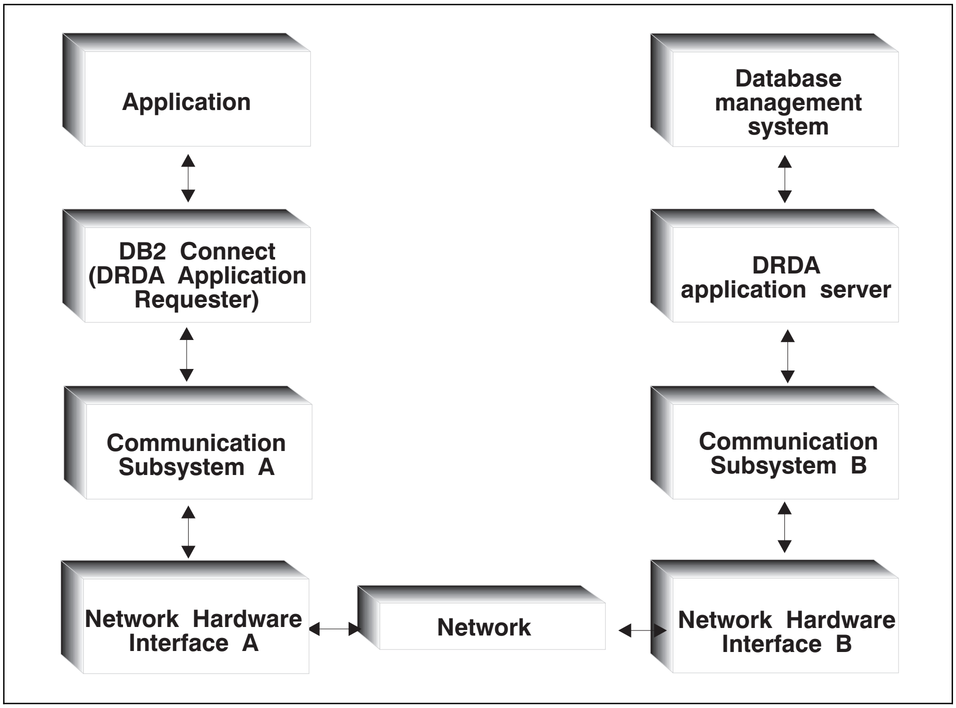
Application
Database management system
DB2 Connect (DRDA Application Requester)
DRDA application server
Communication Subsystem A
Communication Subsystem B
Network Hardware Interface A
Network
Network Hardware Interface B

Figure 8. Data Flows in DB2 Connect

- The DRDA database and part of communication subsystem B are usually running on the same system. This system is made up of one or more CPUs, main storage, an I/O subsystem, DASD, and an operating system. Because other programs may share these components, resource contention could cause performance problems.
- The network is composed of a combination of cables, hubs, communication lines, switches, and other communication controllers. For example, the network hardware interface B could be communication controllers such as 3745 or 3172 or a token ring adapter for an AS/400. There could be more than one transmission medium involved between network hardware interfaces A and B.

- Network hardware interface A could be token ring, Ethernet**, other LAN adapter, or an adapter which supports the SDLC or X.25 protocols. Communication subsystem A might be a product such as IBM Communications Server for OS/2, Microsoft SNA Server, IBM SNA Server for AIX, or SNAplus for HP-UX.
- The DB2 Connect product and the communication subsystem A are usually located on the same system. In this chapter, we assume that the application is also on the same system.

Bottlenecks

Transaction throughput is dependent on the slowest component in the system. If you identify a performance bottleneck, you can often alleviate the problem by changing configuration parameters, allocating more resources to the problem component, upgrading the component, or adding a new component to offload some of the work.

You can use various tools to determine how much time a query spends in each component. This will give you an idea of which components should be tuned or upgraded to improve performance. For example, if you determine that a query spends 60% of its time in the DB2 Connect machine, you might want to tune DB2 Connect or (if you have remote clients) add another DB2 Connect machine to the network.

For more information about performance tools, see “Performance Tools” on page 76.

Benchmarking

Benchmarking is a way to compare performance in one environment with performance in another.

Benchmarking can begin by running the test application in a normal environment. As a performance problem is narrowed down, specialized test cases can be developed to limit the scope of the function that is tested and observed.

Benchmarking does not need to be complex. Specialized test cases need not emulate an entire application in order to obtain valuable information. Start with simple measurements and increase the complexity only when warranted.

Characteristics of good benchmarks (or measurements) include:

- Each test is repeatable.
- Each iteration of a test is started in the same system state.
- The hardware and software used for benchmarking matches your production environment.
- There are no functions or applications active in the system other that those being measured (unless the scenario includes some amount of other activity going on in the system).

  Note: Applications that are started use memory even when they are minimized or idle. This could cause paging and skew the results of the benchmark.

Performance Tools

The following table lists some of the tools that can help you measure system performance. Because these tools themselves use system resources, you might not want to have them active all the time.

Table 6. Performance Tools

System	**Tool**	**Description.**
	CPU and memory usage	
AIX	vmstat, time, ps, tprof	Provide information about CPU or memory contention problems on the DB2 Connect workstation and remote clients
HP-UX	vmstat, time, ps, monitor and glance if available	
OS/2	SPM/2, THESEUS/2, pstat	
Win NT	MS Performance Monitor	
	Database activity	
All	Database monitor	Determines if the problem originates from the database.
MVS or OS/390	DB2PM (IBM), OMEGAMON/DB2 (Candle), TMON (Landmark), INSIGHT (Goal Systems) and DB2AM (BMC)	
Win NT	MS Performance Monitor	
	Network activity	
AIX	netpmon	Reports low level network statistics, including TCP/IP and SNA statistics such as the number of packet or frames received per second.
DOS or OS/2	Token-Ring Network 16/4 Trace and Performance Program	Most network monitors are platform dependent; this tool works for token-ring only.
Network controller such as 3745	NetView Performance Monitor	Reports utilization of communication control and VTAM
OS/2	DatagLANce	A trace tool that presents performance-related data to users graphically.
UNIX-based	netstat	Handles TCP/IP traffic

Application Design

When you create an application, you can improve performance in several ways, including:

- Use compound SQL and stored procedures.
- Group requests.
- Use predicate logic to request only the data that you need.
- Use data blocking.
- Use static SQL whenever possible.

Compound SQL and Stored Procedures

For applications that send and receive many commands and replies, network overhead can be significant. Compound SQL and stored procedures are two ways to reduce this overhead.

If an application sends several SQL statements without intervening programming logic, you can use compound SQL. If programming logic is needed within the group of SQL statements, you can use stored procedures.

All executable statements except the following can be contained within a Compound SQL statement:

```
CALL
FETCH
CLOSE
OPEN
Compound SQL
Connect
Prepare
Release
Describe
Rollback
Disconnect
Set connection
execute immediate
```

See *SQL Reference* for more details.

For information about using compound SQL in an application, see "NOT ATOMIC Compound SQL" on page 55. For information about using compound SQL with the import utility, see "Using Import and Export Utilities" on page 29

For information about using stored procedures, see "Stored Procedures" on page 54.

Grouping Requests

Grouping related database requests (SQL statements) into one database request can reduce the number of requests and responses transmitted across the network. For example, grouping the following statements:

```
SELECT COL1, COL2, COL5, COL6 FROM TABLEA WHERE ROW_ID=1
SELECT COL1, COL2, COL5, COL6 FROM TABLEA WHERE ROW_ID=2
```

into

```
SELECT COL1, COL2, COL5, COL6 FROM TABLEA WHERE ROW_ID=1 OR ROW_ID=2
```

sends fewer requests across the network.

You can also use keywords such as IN and BETWEEN to reduce the number of rows returned. In addition, you can use WHERE, IN, and BETWEEN keywords on UPDATE and DELETE statements.

Predicate Logic

You can use predicate logic to request only the rows and columns that are needed. This minimizes the network traffic and CPU overhead for data transmission.

For example, do not use the query:

```
SELECT * FROM TABLEA
```

if only the first row of TABLEA with ROW_ID=1 is really needed or if only column 1 and column 2 are needed.

Data Blocking

You should use data blocking if you expect large amounts of data from the server. Blocking improves the utilization of the network bandwidth and reduces the CPU overhead of both the DRDA server and the DB2 Connect workstation.

There is fixed amount of CPU and network overhead for each message sent and received regardless of size. Data blocking reduces the number of messages required for the same amount of data transfer.

With blocking, the first row of data from a query will not be delivered to the application until the first *block* is received. Blocking increases the retrieval time for the first row, but improves the retrieval time for subsequent rows.

Another consideration is the amount of memory that is used. The memory working set usually increases when blocking is turned on. For a complete discussion of blocking when using SNA connections, see the *DRDA Connectivity Guide*.

Within DB2 Connect, you can control the amount of data that is transferred within each block, as described in “RQRIOBLK” on page 80.

To invoke blocking, use the BLOCKING option of the prep or bind command. (For more information, see “The BIND Command” on page 24.) Blocking is on, if:

- The cursor is read-only, or
- The cursor is ambiguous and blocking is specified during the prep or bind.

For the definitions of read-only, updateable, and ambiguous cursor, refer to the *Embedded SQL Programming Guide*.

Note: When using dynamic SQL, the cursor is always ambiguous.

SQL Statements with BLOCKING

Updateable SELECT statements (using `UPDATE/DELETE WHERE CURRENT OF` statements) are non-blocking queries, so you should use them only when absolutely necessary.

An updateable SELECT ensures that the row has not changed between the time the `SELECT` is completed and the `UPDATE/DELETE` is issued. If this level of concurrency is not important to your application, an alternative is to use a `DELETE` or `UPDATE` with search criteria based on the values returned from a non-updateable SELECT.

For read-only `SELECT`, specify `FOR FETCH ONLY` (except under VM and VSE, where it is not supported).

Static and Dynamic SQL

Use static SQL as much as possible. It avoids run-time SQL section preparation and ambiguous cursors. If dynamic SQL cannot be avoided, you can do the following to minimize the network traffic and improve performance:

- If the statement is a `SELECT` and must be prepared, perform `PREPARE ... INTO` SQLDA. The SQLDA should be allocated to the full size needed for your settings. If the maximum number of columns is *x* and is expected to stay that way, allocate an SQLDA with *x* SQLVARs. If the number of potential columns is uncertain (and memory is not a problem), use the maximum number of SQLVARs (256).

  If the SQLDA allocation is not big enough to store the returning SQLDA, the program must issue another DESCRIBE with a big enough SQLDA to store the result again. This would increase the network traffic.

  Do not use the PREPARE and DESCRIBE sequence. Using the `PREPARE.....INTO` statement provides better performance.

- Execute statically bound SQL COMMIT or ROLLBACK statements instead of dynamic COMMIT or ROLLBACK statements.
- If it is not a SELECT, COMMIT, or ROLLBACK statement, issue EXECUTE IMMEDIATE to execute the statement instead of the PREPARE and EXECUTE sequence.

Other SQL Considerations

Using the command line processor is, in general, slower than having dynamic SQL in the program because the command line processor must parse the input before submitting the SQL to the database engine. The command line processor also formats data when it is received, which may not be necessary for your application.

SQL statements in an interpreted language (such as REXX) are substantially slower than the same SQL statements in a compiled language (such as C).

There are two types of CONNECT statement, called type 1 and type 2. With type 2 connect, connecting to a database puts the previous connection into a dormant state but does not drop it. If you later switch to a dormant connection, you avoid the overhead of loading libraries and setting up internal data structures. For this reason, using type 2 connect may improve performance for applications that access more than one database. For more information about type 2 connects, see the *Administration Guide* and the *SQL Reference*.

DB2 Connect Tuning

Various parameters in the database manager configuration file can be used to tune DB2 Connect. For information about changing these parameters, see the *Administration Guide*.

RQRIOBLK

The RQRIOBLK parameter sets the maximum size of network I/O blocks. A larger block size may improve the performance of large requests. The block size does not usually affect the response time for small requests, such as a request for a single row of data.

A larger block size usually requires more memory on the DB2 Connect workstation. This increases the size of the working set and may cause large amounts of paging on small workstations.

Use the default DRDA block size (32767) if it does not cause too much paging on executing your application. Otherwise, reduce the I/O block size until there is no paging. Once paging begins, a noticeable degradation of performance will occur. Use performance monitor tools (such as the vmstat tool for UNIX-based systems or SPM/2 for OS/2) to determine whether paging is occurring on your system. For other tools, refer to “Performance Tools” on page 76.

DIR_CACHE

The DIR_CACHE parameter determines whether directory information is cached. With caching (DIR_CACHE=YES), directory files are read and cached in memory to minimize the overhead of creating the internal directory structure and reading the directory files every time a connection is established.

Without caching (DIR_CACHE=NO), whenever you connect to a database the appropriate directory is read from a disk and then the search is performed. After the requested entries are found, all memory related to directory searches is freed.

With caching, a shared directory cache is built during **db2start** processing and freed when DB2 stops. This cache is used by all DB2 server processes (`db2agent`). Also, a private application directory cache is built when an application issues its first connect to a database and freed when the application ends.

Each cache provides an image of the system database directory, the database connection services directory and the node directory. The cache reduces connect costs by eliminating directory file I/O and minimizing directory searches.

If a cached directory is updated, the changes are not immediately propagated to the caches. If a directory entry is not found in a cache, the original directory is searched.

Caching increases the private memory that is needed for the life of an application. Without caching, this memory is needed only when a directory lookup is processed. Overall use of shared memory by DB2 increases slightly because directory information that is shared among database agents is moved to shared memory. The size of the memory required for a cache depends on the number of entries defined in each directory.

Other DB2 Connect Parameters

MAXDARI and NUMDB should be set to their minimum values if there is no local database on the DB2 Connect workstation. This will minimize resource consumption.

AGENTPRI applies only with remote clients. AGENTPRI controls the priority given by the operating system scheduler to agents of a DB2 Connect instance. The DB2 Connect instance is granted more CPU cycles if it has a higher priority (lower number). This reduces the number of CPU cycles left for other processes executing on the DB2 Connect workstation. For example, you could have a high-priority DB2 Connect instance and a low-priority DB2 Connect instance running on the same workstation with different AGENTPRI values.

Every connection from a client machine to a DRDA database through DB2 Connect requires an agent running on the DB2 Connect workstation. Set MAXAGENTS to a value greater than or equal to the peak number of remote client connections accessing a DRDA database through the DB2 Connect workstation.

If you decide to use accounting strings, using the sqlesact() API has performance advantages over the DB2ACCOUNT environment variable method. For more information, see “Implementing Chargeback Accounting” on page 58.

If you do not need a tailored SQLCODE mapping file, you can improve performance by using the default SQLCODE mapping or turning off SQLCODE mapping. (The default mapping file is imbedded in the DB2 Connect library; a tailored mapping file must be read from disk, which affects performance.) For more information about SQLCODE mapping, see Chapter 8, “SQLCODE Mapping” on page 61.

Database Tuning

System performance will be affected by the performance of the DRDA server database.

Different database management systems have different performance features. SQL optimizers of different systems, for example, could behave differently with the same application. Check your DRDA server system performance documentation for more information.

For DB2 for AS/400, you may be able to improve performance by using the uncommitted read (UR) or no commit (NC) bind options to avoid journaling. Please note that when using UR, unjournalled data can only be read, not updated, and then only if blocking is set to ALL.

Depending on the Application Server and the lock granularity it provides, the isolation level used for a query or application may have a significant effect on performance.

The database should have the appropriate level of normalization, effective use of indexes, and suitable allocation of database space. Performance can also be affected by the data types that you use, as described in the following sections.

Data Conversion

When data is transferred from one environment to another, it may need to be converted. This conversion can affect performance.

Consider the following platforms:

- Intel (OS/2)
- IEEE (UNIX-based systems)
- System/370 and System/390 (MVS, OS/390, VM, and VSE)
- OS/400.

and the following types of numeric data:

- Packed decimal
- Zoned decimal
- Integer
- Floating point.

Table 7 on page 83 shows when conversion takes place.

Table 7. Data Conversion

	Intel	**IEEE**	**S/370 & S/390**	**OS/400**
Packed decimal data				
Intel	No	No	No	No
IEEE	No	No	No	No
S/370/390	No	No	No	No
OS/400	No	No	No	No
Zoned decimal data				
Intel	No	No	Yes	Yes
IEEE	No	No	Yes	Yes
S/370/390	Yes	Yes	No	No
OS/400	Yes	Yes	No	No
Integer data				
Intel	No	Yes	Yes	Yes
IEEE	Yes	No	No	No
S/370/390	Yes	No	No	No
OS/400	Yes	No	No	No
Floating point data				
Intel	No	Yes	Yes	Yes
IEEE	Yes	No	Yes	No
S/370/390	Yes	Yes	No	Yes
OS/400	Yes	No	Yes	No

The CPU cost of single-byte character data conversion is generally less than that of numeric data conversion (where data conversion is required).

The data conversion cost of DATE/TIME/TIMESTAMP is almost the same as that of single-byte CHAR. FLOATING point data conversion costs the most. The application designer may want to take advantage of these facts when designing an application based on DB2 Connect.

If a database table has a column defined 'FOR BIT DATA', the character data being transferred between the application and the database does not require any data conversion. This can be used when you are archiving data on the DRDA server.

Data Types for Character Data

Character data can have either the CHAR or VARCHAR data type. Which data type is more efficient depends on the typical length of data in the field:

- If the size of actual data varies significantly, VARCHAR is more efficient because CHAR adds extra blank characters to fill the field. These blank characters must be transmitted across the network like any other characters.
- If the size of actual data does not vary much, CHAR is more efficient because each VARCHAR field has a few bytes of length information which must be transmitted.

Communication System Tuning (SNA)

You should consider the following:

- Size of data transfer blocks

  Increasing the data transfer block size (buffer, message, block, RU (Request Unit), frame, etc.) reduces overhead of data transmission over a period of time. In communication subsystems such as IBM Communications Server for OS/2 and Microsoft SNA Server the RUSIZE parameter controls the data transfer block size.

  The SNA communication system compares the RU size of each partner and uses the smaller of the two RU sizes. You can start tuning with an RU size of 4KB and increase it if necessary. You may get the best performance by setting the RUSIZE and the RQRIOBLK (see "RQRIOBLK" on page 80) to the same value.

- Pacing values

  All APPN-compliant platforms perform adaptive pacing. This means that you do not need to worry about pacing values. If you are using software that is not APPN-compliant (such as VTAM 3.4), you may want to set the values of pacing parameters.

  The most important pacing values to DB2 Connect are the ones utilized at the APPL-APPL layer of the SNA protocol stack:

  – VPACING in the APPL statement in VTAM
  – PACING in the mode definition in Communications Server for OS/2.
  – Send and Receive PACING values in the MODE profile in Microsoft SNA Server.

  A higher pacing value generally improves the data transfer throughput and utilization of the transmission medium.

  In NCP, the other pacing values found in other definitions are to prevent the NCP's resources from being overrun.

- Availability of connections

  Performance depends on keeping connections active and running. For example, you should maintain the SNA connection until other factors (such as cost) become more important.

- Data compression

  If network transmission time is a more important performance factor than CPU processing time, data compression might improve performance. You can use data compression only if it is supported by the communication products at both ends of the connection.

- Communications software tuning

  The tuning of your communications software can affect performance. For more information, see books such as *SNA Server for AIX and SNA Server Gateway Performance Guide*. (This book applies specifically to AIX systems, but the information in it may also be useful to other users.)

Network Hardware

The following considerations relate to the hardware:

- Speed of the network or transmission media

  Performance improves with a faster transmission medium. For example, the following are some typical raw data transfer rates:

Channel-to-channel (fiber optics)	4.0 MB/s
16 Mbps LAN	2.0 MB/s
Channel-to-channel (regular)	1.0 MB/s
4 Mbps LAN	0.5 MB/s
high speed T1 carrier (1.544 Mbps)	0.193 MB/s
fast remote 56 Kbps phone line	0.007 MB/s
19.6 Kbps modem	0.002 MB/s
9600 bps modem	0.001 MB/s

  The data transfer rate is limited by the slowest transmission medium in the path to the DRDA server.

- Network adapter or communication controller

  You should carefully plan the memory usage of the network adapter and communication controller. In addition, you should work with a network specialist to ensure that the controller has the capability to handle the extra traffic generated by DB2 Connect.

- Network topology

  If data crosses from LAN to LAN, and from one SNA Network to another SNA Network, consider the travel time. Bridges, routers, and gateways will add to the elapsed time. For example, reducing the number of bridges that are crossed reduces the number of hops required for each request.

  The physical distance between nodes should also be considered. Even if a message is transferred by satellite, the transfer time is limited by the speed of light (`3 * 10**8 m/s`) and the round-trip distance between the sender and receiver.

- Network traffic

  If the bandwidth of the network has been fully utilized, both the response time and the data transfer rate for a single application will decrease.

  Congestion can occur in the network when data accumulates at a particular part of the network; for example, at an old NCP with a very small buffer size.

- Network reliability

  If the error rate of the network is high, the throughput of the network will decrease and this will cause poor performance because of data re-transmission.

Contention for System Resources

Performance could be degraded if many tasks in the system are contending for system resources. Consider the following questions:

- Is the CPU saturated? Consider upgrading the system, reducing the system workload, and tuning the system to reduce processing overhead.
- Is the memory over-committed? Consider upgrading memory, reducing system workload and tuning the system to reduce the memory working set.
- Is the communication adapter/communication controller too busy? Consider upgrading the network or pairing up token-ring cards.
- Is one of the subsystems too busy, and is this subsystem on the data path?
- Are any unnecessary processes or tasks running on the system? The general rule is not to configure or start services unless they are used regularly since they will waste system resources.
- Do a few processes or tasks use most of the resource? Can they be stopped? Can their priorities be reduced? Can they be refined so that they don't use as much resource?

Performance Troubleshooting

If DB2 Connect users are experiencing long response times during large queries from DRDA AS hosts, the following areas should be examined for the possible cause of the performance problem:

1. For queries which result in returning large data blocks from the DRDA server (usually 32K of data and above), ensure that the database manager configuration parameter RQRIOBLK is set to 32767. This can be done using the Command Line Processor (CLP) as follows:

   ```
   db2 update database manager configuration using RQRIOBLK 32767
   ```

2. If VTAM is used in the connection to DRDA AS, look under "switched major node" configuration for the value of the PACING parameter. On the DB2 Connect workstation, examine the communication setup of the "LU 6.2 Mode Profile" for IBMRDB mode definition. In this definition, ensure the value for the "Receive pacing window" parameter is less than or equal to the PACING value defined on VTAM. A common value for "Receive pacing window" on the DB2 Connect workstation and "PACING" on VTAM is 8.
3. Ensure the maximum RU size defined in the IBMRDB mode definition is set to a suitable value. We recommend not less than 4K for connections using Token-ring hardware. For connections using Ethernet hardware, note the maximum Ethernet frame size of 1536 bytes, which may be a limiting factor.
4. Consult with the VTAM administrator in your environment to ensure that VTAM is using "adaptive pacing" in LU-LU sessions with your DB2 Connect workstation.

Appendix A. Directory Customization Worksheet

Use this worksheet to customize your directories. Refer to “Updating the Directories” on page 17 or the *Command Reference* for command syntax.

Table 8. Node Directory Parameters

Parameter	**Example**	**Your value.**
Node name	DB2NODE or MVSIPNOD	
Symbolic destination name (APPC node)	DB2CPIC	
Remote hostname (TCP/IP node)	MVSHOST	
Server (TCP/IP service name or port number)	db2inst1c (or port number)	
Security type	PROGRAM for APPC Nodes; NONE for TCP/IP nodes.	

Notes:

1. The default TCP/IP port number for DRDA is 446
2. Use SECURITY NONE for TCP/IP nodes unless you know that the DRDA server supports SECURITY SOCKS.

Table 9. DCS Directory Parameters

Parameter	**Example**	**Your value.**
Database name	DB2DB	
Target database name	NEW_YORK3	
Application requester		
Parameter string		

Table 10. System Database Directory Parameters

Parameter	**Example**	**Your value.**
Database name	DB2DB	
Database alias	NYC3	
Node name	DB2NODE	
Authentication	DCS	

Appendix B. National Language Support Considerations

DB2 Connect has the following national language support considerations:

- DB2 Connect messages are translated into certain languages. For information about accessing translated messages, see the *Quick Beginnings* for your platform.
- DB2 Connect supports a large number of languages and code pages. For a list of these code pages, see the *Administration Guide* or the *Road Map to DB2 Programming*.
- When data is transferred between DB2 Connect and a DRDA server, it is usually converted from a workstation code page to a host CCSID (and vice versa). This is described in the remainder of this appendix.

Further information about using DB2 Connect can be found in the DB2 Connect *Quick Beginnings* manuals, including:

- Date and time formats
- Which languages are supported by DB2 Connect Enterprise Edition and DB2 Connect Personal Edition
- How to customize your DB2 Connect workstation for your particular national language environment
- How to customize your host Coded Character Set Identifier (CCSID) setting.

Conversion of Character Data

When character data is transferred between machines, it must be converted to a form that the receiving machine can use.

For example, when data is transferred between the DB2 Connect workstation and a DRDA server, it is usually converted from a workstation code page to a host CCSID, and vice versa. If the two machines use different code pages or CCSIDs, code points are mapped from one code page or CCSID to the other. This conversion is always performed at the receiver.

Character data sent *to* a database consists of SQL statements and input data. Character data sent *from* a database consists of output data. Output data that is interpreted as bit data (for example, data from a column declared with the FOR BIT DATA clause) is not converted. Otherwise all input and output character data is converted if the two machines have different code pages or CCSIDs.

For example, if DB2 Connect is used to access DB2 for OS/390 or DB2/MVS data, the following happens:

1. DB2 Connect sends an SQL statement and input data to OS/390 or MVS.
2. DB2 for OS/390 converts the data to an EBCDIC CCSID and processes it.
3. DB2 for OS/390 sends the result back to the DB2 Connect workstation.
4. DB2 Connect converts the result to an ASCII or ISO code page and returns it to the user.

The table that follows shows the conversions that are supported between code pages (on the workstation) and CCSIDs (on the host).

For more detailed information about supported code page conversions, refer to the *Administration Guide*.

Table 11. Workstation Code Page to Host CCSID Conversion

Host CCSIDs	**Code Page**	**Countries**
037, 273, 277, 278, 280, 284, 285, 297, 500, 871	437, 819, 850, 860, 863, 1004, 1051, 1252, 1275	Australia, Austria, Belgium, Brazil, Canada, Denmark, Finland, France, Germany, Iceland, Italy, Latin America, Netherlands, New Zealand, Norway, Portugal, South Africa, Spain, Sweden, Switzerland, UK, USA
875, 423	813, 869, 1253, 1280	Greece
870	852, 912, 1250, 1282	Croatia, Czech Republic, Hungary, Poland, Romania, Serbia/Montenegro (Latin), Slovakia, Slovenia
1025	855, 915, 1251, 1283	Bulgaria, FYR Macedonia, Serbia/Montenegro (Cyrillic)
1026	857, 920, 1254, 1281	Turkey
424	862, 916, 1255	Israel
420	864, 1046, 1089, 1256	Arabic countries
1025	866, 915, 1251, 1283	Russia
838	874	Thailand
930, 939, 1027, 5026, 5035	932, 942, 943, 954, 5039	Japan
937	938, 948, 950, 964	Taiwan
933	949, 970, 1363	Korea
935	1381, 1383	People's Republic of China

Note that code page 1004 is supported as code page 1252.

In general, data can be converted from a code page to a CCSID and back again to the same code page with no change. The following are the only exceptions to that rule:

- In double-byte character set (DBCS) code pages, some data containing user-defined characters may be lost.
- For single-byte code pages defined within mixed-byte code pages, some characters may be mapped to substitution characters and then lost when the data is converted back to the original code page.

Appendix C. Using DCE Directory Services

With DCE Cell Directory Services (CDS), you can store server information in CDS rather than having to store server information on each client. At present, this is supported for DB2 Universal Database V5.0 clients and DB2 Connect Enterprise Edition V5.0 on AIX, other supported UNIX operating systems, and OS/2.

Note: If you want to use DCE directory services support in DB2 Connect to connect to DB2 for MVS/ESA over SNA connections, then you must apply the DB2 for MVS/ESA PTF UN73393 that supports the use of `DB2DRDA` as the name of the remote transaction program name (RTPN).

If you want to use a DCE directory, you must create the following:

- The database object, which contains information about a database
- The database locator object, which contains information about the connection between remote clients and the DB2 Connect workstation
- The routing information object, which matches database objects to database locator objects

For each DRDA server database that you will access, before creating these objects, you should:

- Make sure that the following DCE attributes have been added to the cds attributes file on the workstation from which you created the objects. On an AIX system, the filename is */etc/dce/cds_attributes*. On an OS/2 system, the filename is *x:\opt\dcelocal\etc\cds_attr*, where *x:* is the drive name.

  ```
  1.3.18.0.2.4.30 DB_Comment                  char
  1.3.18.0.2.4.31 DB_Communication_Protocol   char
  1.3.18.0.2.4.32 DB_Database_Protocol        char
  1.3.18.0.2.4.33 DB_Database_Locator_Name    char
  1.3.18.0.2.4.34 DB_Native_Database_Name     char
  1.3.18.0.2.4.35 DB_Object_Type              char
  1.3.18.0.2.4.36 DB_Product_Name             char
  1.3.18.0.2.4.37 DB_Product_Release          char
  1.3.18.0.2.4.38 DB_Target_Database_Info     char
  1.3.18.0.2.4.39 DB_Authentication           char
  1.3.18.0.2.4.63 DB_Principal                char
  ```

- Make sure you have logged into DCE with sufficient authority to create the objects. The following DCE command can be used to login on a UNIX system:

  ```
  dce_login principal-id password
  ```

  The following DCE command can be used to login on an OS/2 system:

  ```
  dcelogin principal-id password
  ```

Note: Before you can connect to databases using these objects, you should also configure communications on the DRDA server and workstations. This is described in *DB2 Connect Enterprise Edition Quick Beginnings*, and *DB2 Connect Personal Edition Quick Beginnings*.

Creating a Database Object

The database object defines the DRDA server to DB2 Connect. It must always be defined.

For each DRDA server database that you will access, use the DCE command `cdscp create object` to create a database object. For example:

```
   cdscp create object database_global_name
```

Add the following attributes to the object:

DB_Object_Type
D for database

DB_Product_Name
The relational database product (for example, DB2_for_MVS, or DB2_for_OS390)

DB_Native_Database_Name
The database name on the DRDA server system, as follows:

MVS or OS/390	The LOCATION value
VSE or VM	The database name
OS/400	The relational database name

DB_Database_Protocol
DRDA

DB_Authentication
`SERVER`, `CLIENT`, or `DCE`, as described in “Security with DCE Directory Services” on page 97.

DB_Principal
If the Authentication method is DCE, enter the DCE Principal in this attribute.

DB_Communication_Protocol
The following information about the communication protocol between the DB2 Connect workstation and the DRDA server:

1. For the communication protocol `APPC`:

   a. The communication protocol (APPC)

   b. The network ID of the DRDA server database

   c. The LU name for the DRDA server database

   d. The transaction program name for connections to the DRDA server. For DB2 for MVS/ESA, specify DB2DRDA. For any other operating system, specify a valid value that is not in hexadecimal format.

   e. The mode name

f. The security type, as described in “Security with DCE Directory Services” on page 97. For example:

```
APPC;SPIFNET;NYM2DB2;DB2DRD;IBMRDB;PROGRAM
```

2. For the communication protocol `TCPIP`:

   a. The communication protocol (TCPIP)

   b. The destination TCP/IP hostname (for the DRDA server database).

   c. The TCP/IP port number.

   d. Type of connection (whether using `SOCKS` or `NONE`). This is optional. If not specified, `NONE` is used. For example, the following are the attribute values for the communication protocol TCP/IP:

```
tcpip;jaguar;19713;NONE
```

To create a Database object with system security you could put the following instructions into a file:

```
create object /.../cdscell1/subsys/database/DBMVS01
add    object /.../cdscell1/subsys/database/DBMVS01 DB_Object_Type=D
add    object /.../cdscell1/subsys/database/DBMVS01 DB_Product_Name=DB2_for_MVS
add    object /.../cdscell1/subsys/database/DBMVS01 DB_Database_Protocol=DRDA
add    object /.../cdscell1/subsys/database/DBMVS01 DB_Native_Database_Name=\
                   NEW_YORK
add    object /.../cdscell1/subsys/database/DBMVS01 DB_Authentication=SERVER
add    object /.../cdscell1/subsys/database/DBMVS01 DB_Communication_Protocol=\
                   APPC;SPIFNET;NYM2DB2;DB2DRDA;IBMRDB;PROGRAM
```

and then enter the command:

```
cdscp < filename
```

Note: In the file, specify a backslash (\) whenever you want a statement to continue to the next line.

To create a Database object with DCE security you could put the following instructions into a file:

```
create object /.../cdscell1/subsys/database/DBMVS02
add    object /.../cdscell1/subsys/database/DBMVS02 DB_Object_Type=D
add    object /.../cdscell1/subsys/database/DBMVS02 DB_Product_Name=DB2_for_MVS
add    object /.../cdscell1/subsys/database/DBMVS02 DB_Database_Protocol=DRDA
add    object /.../cdscell1/subsys/database/DBMVS02 DB_Native_Database_Name=\
                   NEW_YORK
add    object /.../cdscell1/subsys/database/DBMVS02 DB_Authentication=DCE
add    object /.../cdscell1/subsys/database/DBMVS02 DB_Principal=\
               /.../cdscell1/principal_name
add    object /.../cdscell1/subsys/database/DBMVS02 DB_Communication_Protocol=\
                   APPC;SPIFNET;NYM2DB2;DB2DRDA;IBMRDB;NONE
```

and then enter the command:

```
cdscp < filename
```

Creating a Database Locator Object

The database locator object is used to define a DB2 Connect Enterprise Edition gateway to its clients.

For your DB2 Connect workstation, use the DCE command `cdscp create object` to create a database locator object. For example:

```
cdscp create object object_global_name
```

Add the following attributes to the object:

DB_Object_Type
L for locator object

DB_Communication_Protocol
The following information about each communication protocol between the DB2 Connect workstation and remote clients.

For APPC:

1. The communication protocol (`APPC`)
2. The network ID of the DB2 Connect workstation
3. The LU name for the DB2 Connect workstation
4. The transaction program name for connections from remote clients
5. The mode name
6. The security type, as described in “Security with DCE Directory Services” on page 97.

For TCP/IP:

1. The communication protocol (`TCPIP`)
2. The host name of the DB2 Connect workstation
3. The connection port used by the DB2 Connect workstation to accept connections from remote clients
4. Type of connection (whether using SOCKS or NONE). This is optional. If not specified, NONE is used.

For IPX/SPX:

1. The communication protocol (`IPXSPX`)
2. The file server name. Use `*` for direct addressing
3. The object name. Use the internetwork address for direct addressing.

For NETBIOS:

1. The communication protocol (`NETBIOS`)
2. The `NNAME` for the server or DB2 Connect Enterprise Edition gateway.

For example, you could put the following lines into a file:

```
create object /.../cdscell1/subsys/database/DBAIX01
add    object /.../cdscell1/subsys/database/DBAIX01 DB_Object_Type= L
add    object /.../cdscell1/subsys/database/DBAIX01 DB_Communication_Protocol=\
          TCPIP;AIX001;3700
add    object /.../cdscell1/subsys/database/DBAIX01 DB_Communication_Protocol=\
          APPC;SPIFNET;NYX1GW01;NYSERVER;IBMRDB;NONE
```

and then enter the command:

```
cdscp < filename
```

On OS/2, you can also specify the IPXSPX or NETBIOS protocol in the DB_Communication_Protocol attribute. For example:

```
 - IPXSPX;fileserver;objectname
 - NETBIOS;nname
```

Creating a Routing Information Object

This object is required to be defined in DCE and it is retrieved by the DB2 client.

Use the DCE command `cdscp create object` to create a routing information object. For example:

```
cdscp create object object_global_name
```

Add a DB_Object_Type attribute of R.

For each database object, add one DB_Target_Database_Info attribute. Each DB_Target_Database_Info attribute consists of the following parameters:

Database
The name of a database object, including the full path. Specify *OTHERDBS to indicate all other databases that are not specified explicitly.

Outbound protocol
The database protocol for DRDA server connections (DRDA)

Inbound protocol
The database protocol for remote client connections (DB2RA),

Authenticate at Gateway
0 (for No) or 1 (for Yes), as described in “Security with DCE Directory Services” on page 97.

Parameter String for Gateway
The string containing the parameters to be used in the gateway. Its content is gateway specific. For DB2 Connect gateway specific strings refer to “DCS Directory” on page 14.

Database Locator
The name of a database locator object representing the DB2 Connect workstation

For example, you could put the following lines into a file:

```
create object /.../cdscell1/subsys/database/ROUTE1
add    object /.../cdscell1/subsys/database/ROUTE1 DB_Object_Type=R
add    object /.../cdscell1/subsys/database/ROUTE1 DB_Target_Database_Info=\
/.../cdscell1/subsys/database/DBMVS01;DRDA;DB2RA;0;;\
/.../cdscell1/subsys/database/DBAIX01
add    object /.../cdscell1/subsys/database/ROUTE1 DB_Target_Database_Info=\
*OTHERDBS;DRDA;DB2RA;0;;\
/.../cdscell1/subsys/database/DBAIX02
```

and then enter the command:

```
cdcsp < filename
```

Setting Configuration Parameters

Update the Database Manager configuration of the client as follows:

```
DB2 UPDATE DATABASE MANAGER CONFIGURATION USING
[DIR_PATH_NAME path]
DIR_OBJ_NAME loc_obj
DIR_TYPE DCE
[ROUTE_OBJ_NAME route_obj]
[DFT_CLIENT_COMM protocol]
[DFT_CLIENT_ADPT 0-15]
```

where:

- *path* is the default path used to form the complete name of target databases (default `/.:/subsys/database/`)
- *loc_obj* is used to identify the client in the DCE name space
- DIR_TYPE DCE specifies that DCE directories are used by the client application
- *route_obj* is the name of the routing information object (e.g. ROUTE1).
- *protocol* is the communication protocol between the client and the DB2 Connect workstation (APPC or TCPIP for UNIX; APPC, IPXSPX, NETBIOS, or TCP/IP for OS/2).
- Default Client Adapter 0 through 15 for NETBIOS. If the protocol is NETBIOS and the client adapter number is not the default value 0, specify the client adapter number.

Note: The following environment variables can respectively overwrite those listed above.

- DB2DIRPATHNAME can overwrite DIR_PATH_NAME
- DB2ROUTE can overwrite ROUTE_OBJ_NAME
- DB2CLIENTCOMM can overwrite DFT_CLIENT_COMM
- DB2CLIENTADPT can overwrite DFT_CLIENT_ADPT

Cataloging the Database

If a database is in a different path than the default, or if you want to use an alias that is different than the database name, you can catalog the global database. You can use the command line processor CATALOG GLOBAL DATABASE command as follows:

```
db2 CATALOG GLOBAL DATABASE database_global_name
    AS alias
    USING DIRECTORY DCE
```

The alias will be used by any application program that accesses the database.

For example:

```
db2 CATALOG GLOBAL DATABASE /.../cdscell2/subsys/database/dbmvs12 AS NYC3
    USING DIRECTORY DCE
```

Security with DCE Directory Services

As DB2 Connect administrator, you can determine where user names and passwords are validated. With DCE directories, you do this by setting the following:

- The security type of the communication protocol in the database locator object representing the DB2 Connect workstation. Use security type NONE.
- The authentication type of the database object.
- The security type of the communication protocol in the database object
- The `authenticate at gateway` parameter in the routing information object

Table 12 and Table 13 on page 98 show the possible combinations of these values and where validation is performed for each combination. Only the combinations shown in these tables are supported by DB2 Connect with DCE Directory Services.

Table 12. Valid Security Scenarios with DCE using APPC connections

	Database object of the Server		**Routing object**	
Case	**Authentication**	**Security**	**Authentication at DB2 Connect Gateway (1=true, 0=false)**	**Validation**
1	CLIENT	SAME	0	Remote client (or DB2 Connect workstation)
2	CLIENT	SAME	1	DB2 Connect workstation
3	SERVER	PROGRAM	0	DRDA server
4	SERVER	PROGRAM	1	DB2 Connect workstation and DRDA server
5	DCE	NONE	NOT APPLICABLE	At the DCE security server

Note: If a remote client is connected to the DB2 Connect Enterprise Edition gateway workstation via an APPC connection, specify a security type of NONE in the DCE locator object of the gateway.

Table 13. Valid Security Scenarios with DCE using TCP/IP connections

Case	Database object of the Server	Routing object	Validation
	Authentication	**Authentication at DB2 Connect Enterprise Edition Gateway (1=true, 0=false)**	
1	CLIENT	0	Remote client (or DB2 Connect workstation)
2	CLIENT	1	DB2 Connect workstation
3	SERVER	0	DRDA server
4	NOT APPLICABLE	NOT APPLICABLE	None
5	DCE	NOT APPLICABLE	At the DCE security server

Each combination is described in more detail below:

- In the first case, the user name and password are validated only at the remote client. (For a local client, the user name and password are validated only at the DB2 Connect workstation.)

  The user is expected to be authenticated at the location he or she first signs on to. The user ID is sent across the network, but not the password. Use this type of security only if all client workstations have adequate security facilities.

- In the second case, the user name and password are validated at the DB2 Connect workstation only. The password is sent across the network from the remote client to the DB2 Connect workstation but not to the DRDA server.

- In the third case, the user name and password are validated at the DRDA server only. The password is sent across the network from the remote client to the DB2 Connect workstation and from the DB2 Connect workstation to the DRDA server.

- In the fourth case, the user name and password are validated at both the DB2 Connect workstation and the DRDA server. The password is sent across the network from the remote client to the DB2 Connect workstation and from the DB2 Connect workstation to the DRDA server.

  Because validation is performed in two places, the same set of user names and passwords must be maintained at both the DB2 Connect workstation and the DRDA server.

- In the fifth case, a DCE ticket is obtained from the DCE security server.

Notes:

1. For AIX systems, all users using security type `SAME` must belong to the AIX **system** group.
2. For AIX systems with remote clients, the instance of the DB2 Connect product running on the DB2 Connect workstation must belong to the AIX **system** group.
3. Access to a DRDA server is controlled by its own security mechanisms or subsystems; for example, the Virtual Telecommunications Access Method (VTAM) and Resource Access Control Facility (RACF). Access to protected database objects is controlled by the SQL **GRANT** and **REVOKE** statements.

Appendix D. Binding Utilities for Back-Level Clients

If you have remote clients from a previous release, you may need to bind utilities on these clients to the DRDA server:

- If the old client was used with a previous release of DB2 Connect against the same DRDA server, you do not need to do any additional steps.
- If the old client was not used with DB2 Connect (for example, if several OS/2 machines were connected with no connection to a DRDA server), do the following:
  1. If you have any DB2 for OS/2 Version 1.0 or 1.2 clients, create a bind list file with the following lines:

     ```
     sqlabind.bnd+
     sqlueiwi.bnd+
     sqluigsi.bnd+
     sqluiici.bnd+
     sqluiict.bnd+
     sqluexpm.bnd+
     sqluimpm.bnd+
     sqlurexp.bnd+
     sqlarxcs.bnd+
     sqlarxrr.bnd+
     sqlarxur.bnd
     ```

     and copy each of these bind files from one of your clients to the DB2 Connect workstation.
  2. If you have Client Application Enabler Version 1.0 or 1.2, create a bind list file with the following lines:

     ```
     db2ajgrt.bnd+
     db2clics.bnd+
     db2clpcs.bnd+
     db2clprr.bnd+
     db2clpur.bnd+
     db2ueiwi.bnd+
     db2uigsi.bnd+
     db2uiici.bnd+
     db2uiict.bnd+
     db2uexpm.bnd+
     db2uimpm.bnd+
     db2urexp.bnd
     ```

     and copy each of these bind files from one of your clients to the DB2 Connect workstation.
  3. At the DB2 Connect workstation, bind each bind list file to each DRDA server database. Issue commands similar to the following:

```
db2 connect to DBALIAS user USERID using PASSWORD
db2 bind path@bindfile.lst blocking all
      sqlerror continue messages bindfile.msg grant public
db2 connect reset
```

where *DBALIAS*, *USERID*, and *PASSWORD* apply to the DRDA server database, *bindfile* is the name of the bind list file, and *path* is the location of the bind list file.

You can use the `grant` option of the `bind` command to grant EXECUTE privilege to PUBLIC or to a specified user name or group ID. If you do not use the `grant` option of the `bind` command, you must GRANT EXECUTE (RUN) individually for each package.

To find out the package names for the bind files, enter the following command:

```
ddcspkgn @bindfile.lst
```

Appendix E. Notices

Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter 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 the

IBM Director of Licensing,
IBM Corporation,
500 Columbus Avenue,
Thornwood, NY, 10594
USA.

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 Canada Limited
Department 071
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

This publication may contain 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.

Trademarks

The following terms are trademarks or registered trademarks of the IBM Corporation in the United States and/or other countries:

ACF/VTAM
ADSTAR
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
CICS
C Set++
C/370
DATABASE 2
DatagLANce
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
Distributed Relational Database Architecture
DRDA
Extended Services
FFST
First Failure Support Technology
IBM
IMS
Lan Distance
MVS/ESA
MVS/XA
NetView
OS/400
OS/390
OS/2
PowerPC
QMF
RACF
RISC System/6000
SAA
SP
SQL/DS
SQL/400
S/370
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WIN-OS/2

Trademarks of Other Companies

The following terms are trademarks or registered trademarks of the companies listed:

C-bus is a trademark of Corollary, Inc.

HP-UX is a trademark of Hewlett-Packard.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Solaris is a trademark of Sun Microsystems, Inc.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, or service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Index

Contacting IBM

This section lists ways you can get more information from IBM.

If you have a technical problem, please take the time to review and carry out the actions suggested by the *Troubleshooting Guide* before contacting DB2 Customer Support. Depending on the nature of your problem or concern, this guide will suggest information you can gather to help us to serve you better.

For information or to order any of the DB2 Universal Database products contact an IBM representative at a local branch office or contact any authorized IBM software remarketer.

Telephone

If you live in the U.S.A., call one of the following numbers:

- 1-800-237-5511 to learn about available service options.
- 1-800-IBM-CALL (1-800-426-2255) or 1-800-3IBM-OS2 (1-800-342-6672) to order products or get general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, see Appendix A of the IBM Software Support Handbook. You can access this document by selecting the "Roadmap to IBM Support" item at: http://www.ibm.com/support/.

Note that in some countries, IBM-authorized dealers should contact their dealer support structure instead of the IBM Support Center.

World Wide Web
http://www.software.ibm.com/data/
http://www.software.ibm.com/data/db2/library/

The DB2 World Wide Web pages provide current DB2 information about news, product descriptions, education schedules, and more. The DB2 Product and Service Technical Library provides access to frequently asked questions, fixes, books, and up-to-date DB2 technical information. (Note that this information may be in English only.)

Anonymous FTP Sites
ftp.software.ibm.com

Log on as anonymous. In the directory /ps/products/db2, you can find demos, fixes, information, and tools concerning DB2 and many related products.

Internet Newsgroups
comp.databases.ibm-db2, bit.listserv.db2-l

These newsgroups are available for users to discuss their experiences with DB2 products.

CompuServe
GO IBMDB2 to access the IBM DB2 Family forums

All DB2 products are supported through these forums.

To find out about the IBM Professional Certification Program for DB2 Universal Database, go to http://www.software.ibm.com/data/db2/db2tech/db2cert.html

Part Number: 10J8163

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

10J8163

S10J-8163-00