IBM® DB2® Connect

User's Guide

Version 7

SC09-2954-00

IBM® DB2® Connect

User's Guide

Version 7

SC09-2954-00

Before using this information and the product it supports, be sure to read the general information under "Appendix H. Notices" on page 219.

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

Contents

About this Book

This book contains general usage information about the following IBM DB2 Connect products:

- DB2 Connect Personal Edition for OS/2 and Windows 32-bit operating systems.
- DB2 Connect Enterprise Edition (EE) for AIX, HP-UX, Linux, PTX, Solaris, OS/2, and Windows 32-bit operating systems.
- DB2 Connect Unlimited Edition for OS/390.

The DB2 Connect User's Guide is divided into three parts:

- Part 1. An Introduction to DB2 Connect, which provides a conceptual overview of DB2 Connect, Distributed Relational Database Architecture (DRDA), and possible usage scenarios.
- Part 2. Reference and Troubleshooting, which provides information on updating database directories, binding applications, administration utilities, DB2 System Monitor, security, problem determination, and performance.
- Part 3. Appendixes, which provide miscellaneous information, hints, and tips.

This book also explains concepts that are applicable to all DB2 Connect products. For information about a specific platform refer to either:

- *DB2 Connect Personal Edition Quick Beginnings*, for single-user DB2 Connect setup on OS/2 and Windows 32-bit operating systems.
- *DB2 Connect Personal Edition for Linux Quick Beginnings*, for single-user DB2 Connect setup on Linux.
- *DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings*, for multi-user DB2 Connect gateway setup on OS/2 or Windows 32-bit operating systems.
- *DB2 Connect Enterprise Edition for UNIX Quick Beginnings*, for multi-user DB2 Connect gateway setup on AIX, HP-UX, Linux, PTX, or Solaris.

Who Should Read this Book

This book is intended for programmers and administrators who are responsible for setting up and maintaining DB2 Connect connections. These connections can exist between DB2 clients and any of the following Distributed Relational Database Architecture (DRDA) application server database management systems:

- DB2 Universal Database for OS/390 Version 5 or higher
- DB2 for MVS Version 3 or higher
- DB2 for VSE & VM
- DB2 Universal Database for AS/400
- Any other relational database management system that implements a DRDA application server function.

Notes:

1. DB2 Universal Database (DB2 UDB) does not require DB2 Connect to allow host or AS/400 applications to access DB2 UDB data.
2. DB2 Universal Database for OS/390 Version 5.1 or higher is required to use DRDA Level 3 functions, including TCP/IP database connections, and stored procedures with multi-row answer sets.
3. DB2 Universal Database for OS/390 Version 6.1 or higher is required to use the DRDA Level 4 functions for which support is provided in DB2 Connect. These functions include support for big integer, large object, row ID, and user-defined distinct data types.

Part 1. An Introduction to DB2 Connect

Chapter 1. DB2 Connect Overview

DB2 Connect provides extremely fast and robust connectivity to IBM mainframe databases for e-business and other applications running under various UNIX and non-UNIX operating systems.

DB2 Connect has several connection solutions. DB2 Connect Personal Edition provides direct connectivity to host or AS/400 databases, while DB2 Connect Enterprise Edition provides indirect connectivity that allows clients to access host or AS/400 databases through the DB2 Connect server. DB2 Connect Unlimited Edition provides a unique packing solution that makes product selection and licensing easier.

DB2 Connect Enterprise Edition

DB2 Connect Enterprise Edition is a connectivity server that concentrates and manages connections from multiple desktop clients and web applications to DB2 database servers running on host or AS/400 systems. IBM's DB2 for AS/400, DB2 for OS/390, and DB2 for VSE & VM databases continue to be the systems of choice for managing most critical data for the world's largest organizations. While these host and AS/400 databases manage the data, there is a great demand to integrate this data with applications running on Windows, UNIX, and OS/2 workstations.

DB2 Connect Enterprise Edition enables local and remote client applications to create, update, control, and manage DB2 databases and host systems using Structured Query Language (SQL), DB2 APIs (Application Programming Interfaces), ODBC (Open Database Connectivity), JDBC (Java Database Connectivity), SQLJ (Embedded SQLJ for Java), or DB2 CLI (Call Level Interface). In addition, DB2 Connect supports Microsoft Windows data interfaces such as ActiveX Data Objects (ADO), Remote Data Objects (RDO), and OLE DB.

DB2 Connect Enterprise Edition is currently available for AIX, HP-UX, Linux, OS/2, PTX, Solaris, and Windows 32-bit operating systems. These servers provide support for applications running on OS/2, UNIX (AIX, HP-UX, Linux, PTX, Solaris, Silicon Graphics IRIX), and Windows 32-bit workstations.

DB2 Connect Personal Edition

DB2 Connect Personal Edition provides access from a single workstation to DB2 databases residing on servers such as MVS/ESA, OS/390, OS/400, VM and VSE, as well as to DB2 Universal Database servers on OS/2, UNIX, and Windows 32-bit operating systems. DB2

Connect Personal Edition provides the same rich set of APIs as DB2 Connect Enterprise Edition, and also features integrated SNA support on all Windows platforms.

This product is currently available for OS/2, Linux, and Windows 32-bit operating systems.

DB2 Connect Unlimited Edition

DB2 Connect Unlimited Edition is a unique package offering that allows complete flexibility of DB2 Connect deployment and simplifies product selection and licensing. This product contains both DB2 Connect Personal Edition and DB2 Connect Enterprise Edition with license terms and conditions that allow the unlimited deployment of any DB2 Connect product. License charges are based on the size of the System/390 that DB2 Connect users will be working with.

This new package offering is only available for OS/390 systems and licensing is only valid for DB2 for OS/390 data sources.

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 can also refer to:

MVS (Version 4 and earlier)
A DB2 for MVS/ESA subsystem identified by its LOCATION NAME.

The LOCATION NAME can be determined by logging in to TSO and issuing the following SQL query using one of the available query tools:

```
select current server from sysibm.sysdummy1
```

LOCATION NAME is also defined in the Boot Strap Data Set (BSDS) and is provided in the DSNL004I message (LOCATION=location), which is written when the Distributed Data Facility (DDF) is started.

OS/390 (Version 5 and later)
A DB2 Universal Database for OS/390 subsystem identified by its LOCATION NAME.

The LOCATION NAME can be determined by logging into TSO and issuing the following SQL query using one of the available query tools:

```
select current server from sysibm.sysdummy1
```

LOCATION NAME is also defined in the Boot Strap Data Set (BSDS) as well as the DSNL004I message (LOCATION=location), which is written when the Distributed Data Facility (DDF) is started.

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 Universal Database 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. This name is known as the Relation Database Name (RDB Name).

To display the RDB Name of your AS/400 system execute the command **WRKRDBDIRE** on your AS/400. The RDB Name of your Local system has *LOCAL specified in the Remote Location column. To change the RDB Name use the command CHGRDBDIRE.

Setting up DB2 Connect

Before you can use DB2 Connect, you must perform the following steps:

Step 1. Install DB2 Connect, and configure both the host or AS/400 server, and workstation communications, as described in the appropriate *DB2 Connect Quick Beginnings* book or in the *Installation and Configuration Supplement*.

Step 2. Update the database directories as described in "Chapter 6. Updating Database Directories" on page 73.

Note: On OS/2, and Windows 32-bit operating systems, we recommended that you use the Client Configuration Assistant (CCA).

On all other platforms, the database directories must be updated using the DB2 Command Line Processor (CLP). Both approaches are described in the *Installation and Configuration Supplement*.

Step 3. Bind the DB2 Connect utilities to each host or AS/400 database management system, as described in "Chapter 7. Binding Applications and Utilities" on page 87.

This task can also be performed using the CCA or the Data Sources Setup dialog where provided.

DB2 Connect and SQL

DB2 Connect forwards SQL statements submitted by application programs to host or AS/400 database servers. DB2 Connect can forward almost any valid SQL statement. The exceptions are documented in "Host or AS/400 Server SQL Statements Rejected by DB2 Connect" on page 54.

Two types of embedded SQL processing exist: static SQL and dynamic SQL. Static SQL minimizes the time required to execute an SQL statement by processing in advance. Dynamic SQL is processed when the SQL statement is submitted to the host or AS/400 database server. Dynamic SQL is 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 host or AS/400 database servers implement SQL differently. For more information about common SQL statements that are supported by all IBM systems, refer to the *SQL Reference*.

DB2 Connect fully supports the common IBM SQL, as well as the DB2 Universal Database for OS/390, DB2 for MVS/ESA, DB2 for VSE & VM (formerly SQL/DS), and DB2 Universal Database for AS/400 implementations of SQL. IBM SQL is strongly recommended for maintaining database independence. For more information, see "Chapter 4. Programming in a DB2 Connect Environment" on page 41.

Administration Utilities

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

- The Command Line Processor lets you issue SQL statements against a host or AS/400 database server database. It flows the SQL statements to the database that you specify.
- The DB2 Command Center provides a graphical interface to the Command Line Processor.
- Import and export utilities let you load, import, and export data to and from a file on a workstation and a host or AS/400 database server database. These files can then be used for importing data into databases, spreadsheets, and other applications running on your workstation. For more information on import and export utilities, refer to the *Data Movement Utilities Guide and Reference*.
- Users of DB2 Connect Enterprise Edition running on Windows NT and Windows 2000 can use the Event Viewer and the Performance Monitor. Using the Event Viewer, you can view exception events logged by DB2 Connect. Using the Performance Monitor, you can monitor and manage the performance of DB2 Connect servers either locally or remotely.

- The DB2 Control Center lets you administer and monitor all aspects of DB2 Connect servers. It also allows administrators to work with DB2 for OS/390 database objects, such as tables, views, buffer pools, and threads. For more information on managing DB2 for OS/390 systems from the DB2 Control Center, refer to the *Application Development Guide*.

For more information about these utilities, see "Chapter 9. Administration Utilities" on page 107.

In addition, the database system monitor utility lets the system administrator monitor system connections. This utility 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 host or AS/400 database server. For more information, see "Chapter 8. Database System Monitor" on page 95.

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.

In discussing DB2 Connect, we will often refer to the term unit of work. A *unit of work (UOW)* 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.

Another key concept is distributed unit of work, also known as multisite update. A *distributed unit of work (DUOW)* involves more than one database server within a unit of work. We define *multisite update* more specifically as a transaction having the following characteristics:

- More than one database management server is updated per unit of work.
- The application directs the distribution of work, and initiates commit.
- There may be multiple requests per unit of work.
- There is one database management server per request.
- Commitment is coordinated across multiple database servers.

For more information on multisite update, see "Enabling Multisite Updates (Two-Phase Commit)" on page 13.

DRDA and DB2 Connect

DB2 Connect implements the DRDA architecture to reduce the cost and complexity of accessing data stored in DB2 Universal Database for AS/400, DB2 Universal Database 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 (AR)* is the code that handles the application end of a distributed connection; it is the application that is requesting data. An *application server (AS)* is the code that handles the

database end of the connection. In the DB2 Connect environment, the DB2 Connect workstation can only function as an application requester on behalf of application programs.

Figure 1 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.

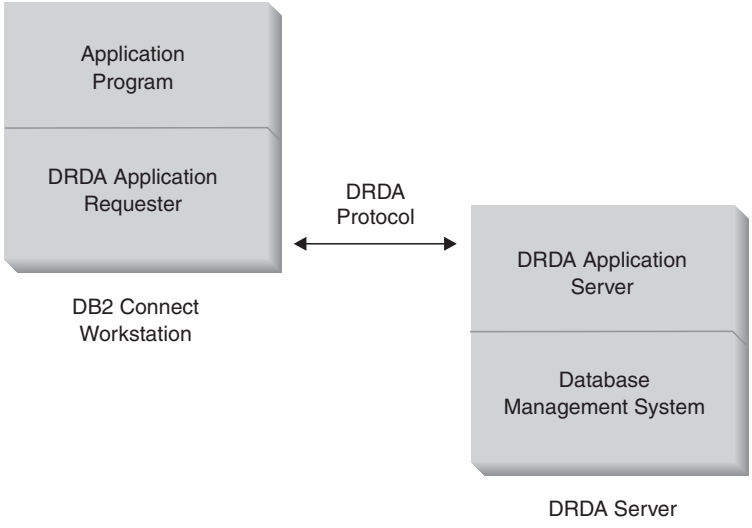


Figure 1. Data Flow Between a DB2 Connect Workstation and a DRDA Server

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 architectures used as 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 *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 update 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.

For example, Figure 2 shows a database client running a funds transfer application that accesses a database containing checking and savings account tables, as well as a banking fee schedule. The application must:

- Accept the amount to transfer from the user interface.
- Subtract the amount from the savings account, and determine the new balance.
- Read the fee schedule to determine the transaction fee for a savings account with the given balance.
- Subtract the transaction fee from the savings account.
- Add the amount of the transfer to the checking account.
- Commit the transaction (unit of work).

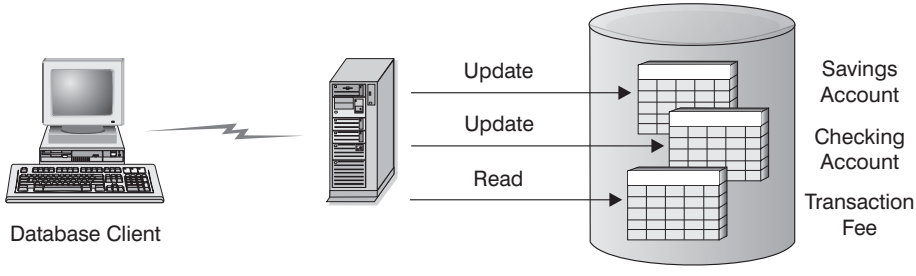


Figure 2. Using a Single Database in a Transaction

To set up such an application, you must:

1. Create the tables for the savings account, checking account and banking fee schedule in the same database as described in the *Administration Guide*.

2. If physically remote, set up the database server to use the appropriate communications protocol, as described in the *Quick Beginnings* books
3. If physically remote, catalog the node and the database to identify the database on the database server, as described in the *Quick Beginnings* books
4. Precompile your application program to specify a type 1 connection; that is, specify CONNECT(1) on the PREP command, as described in the *Application Development Guide*.

Distributed Request

A *distributed request* is a distributed database function that allows applications and users to submit SQL statements that reference two or more DBMSs or databases in a single statement. For example, a join between tables in two different DB2 for OS/390 subsystems.

DB2 Connect Version 7 provides support for distributed requests across databases and DBMSs. You can, for example, perform a UNION operation between a DB2 table and an Oracle view. Supported DBMSs include members of the DB2 Family (such as DB2 UDB for Windows, UNIX and OS/2, DB2 for OS/390 and DB2 for AS/400), and Oracle.

Distributed request provides *location transparency* for database objects. If information (in tables and views) is moved, references to that information (called *nicknames*) can be updated without any changes to applications that request the information. Distributed request also provides *compensation* for DBMSs that do not support all of the DB2 SQL dialect, or certain optimization capabilities. Operations that cannot be performed under such a DBMS (such as recursive SQL) are run under DB2 Connect.

Distributed request function in a *semi-autonomous* manner. For example, DB2 queries containing references to Oracle objects can be submitted while Oracle applications are accessing the same server. Distributed request does not monopolize or restrict access (beyond integrity and locking constraints) to Oracle or other DBMS objects.

Implementation of the distributed request function consist of a DB2 Connect Version 7 instance, a database that will serve as the federated database, and one or more remote data sources. The *federated database* contains catalog entries identifying data sources and their characteristics. A *data source* consists of a DBMS and data. Applications connect to the federated database just like any other DB2 database. DB2 Connect federated database is not licensed for managing user data. Its sole purpose is to contain information about data sources.

After a federated system is set up, the information in data sources can be accessed as though it were in one large database. Users and applications send queries to one federated database, which then retrieves data from DB2 Family and Oracle systems as needed. User and applications specify nicknames in queries; these nicknames provide references to tables and views located in data sources. From an end-user perspective, nicknames are similar to aliases.

Many factors can affect the performance of distributed requests. The most critical factor is to ensure that accurate and up-to-date information about data sources and their objects is stored in the federated database global catalog. This information is used by the DB2 optimizer, and can affect decisions to push down operations for evaluation at data sources. For more information about federated system performance, refer to the *Administration Guide: Performance*.

Enabling Multisite Updates (Two-Phase Commit)

Multisite update, also known as distributed unit of work (DUOW) and two-phase commit, is a function that enables your applications to update data in multiple remote database servers with guaranteed integrity. For example, a banking transaction that involves the transfer of money from one account to another in a different database server.

In such a transaction, it is critical that updates which implement debit operations on one account do not get committed unless updates required to process credits to the other account are committed as well. The multisite update considerations apply when data representing these accounts is managed by two different database servers.

DB2 products provide comprehensive support for multisite updates. This support is available for applications developed using regular SQL as well as applications that use transaction monitor (TP monitor) products that implement the X/Open XA interface specification. Examples of such TP monitors products include IBM TxSeries (CICS and Encina), IBM Message and Queuing Series, IBM Component Broker Series, IBM San Francisco Project as well as Microsoft Transaction Server (MTS), BEA Tuxedo and several others. There are different setup requirements depending on whether native SQL multisite update or TP monitor multisite update is used.

Both the native SQL and TP monitor multisite update programs must be precompiled with the `CONNECT 2 SYNCPOINT TWOPHASE` options. Both can use the SQL Connect statement to indicate which database they want to be used for the SQL statements that follow. If there is no TP monitor to tell DB2 it is going to coordinate the transaction (as indicated by DB2 receiving the `xa_open` calls from the TP monitor to establish a database connection), then the DB2 software will be used to coordinate the transaction.

When using TP monitor multisite update, the application must request commit or rollback by using the TP monitor's API, for example `CICS SYNCPOINT, Encina Abort(), MTS SetAbort()`.

When using native SQL multisite update, the normal `SQL COMMIT` and `ROLLBACK` must be used.

TP monitor multisite update can coordinate a transaction that accesses both DB2 and non-DB2 resource managers such as Oracle, Informix or SQLServer. Native SQL multisite update is used with DB2 servers only.

For a multisite update transaction to work, each of the databases participating in a distributed transaction must be capable of supporting distributed unit of work. Currently, the following DB2 servers provided DUOW support that enabled them to participate in distributed transactions:

- DB2 UDB for UNIX, OS/2, and Windows V5 or later
- DB2 for MVS/ESA V3.1 and 4.1
- DB2 for OS/390 V5.1
- DB2 Universal Database for OS/390 V6.1 or later
- DB2/400 V3.1 or later (SNA only)
- DB2 Server for VM and VSE V5.1 or later (SNA only)
- Database Server 4

A distributed transaction can update any mix of supported database servers. For example, your application can update several tables in DB2 Universal Database on Windows NT or Windows 2000, a DB2 for OS/390 database, and a DB2/400 database, all within a single transaction.

Host and AS/400 Multisite Update Scenarios that Require SPM

Host and AS/400 database servers require DB2 Connect to participate in a distributed transaction originating from PC, UNIX, and Web applications. In addition, many of the multisite update scenarios that involve host and AS/400 database servers require that the Syncpoint Manager (SPM) component be configured. When a DB2 instance is created, the DB2 SPM is automatically configured with default settings.

The need for SPM is dictated by the choice of protocol (SNA or TCP/IP) and use of a TP monitor. The following table provides a summary of scenarios that require the use of SPM. The table also shows that DB2 Connect is required for any access to the host or AS/400 from Intel or UNIX machines. In addition, for multisite updates, the SPM component of DB2 Connect is required if the access is via SNA or uses a TP monitor.

Table 1. Host and AS/400 multisite update scenarios that require SPM

TP Monitor Used?	Protocol	SPM Needed?	Product Required (choose One)	Host and AS/400 Database Supported
Yes	TCP/IP	Yes	• DB2 Connect Enterprise Edition • DB2 Universal Database Enterprise Edition • DB2 Universal Database Enterprise - gExtended Edition	• DB2 for OS/390 V5.1 • DB2 Universal Database for OS/390 V6.1 or later
Yes	SNA	Yes	• DB2 Connect Enterprise Edition* • DB2 Universal Database Enterprise Edition* • DB2 Universal Database Enterprise - Extended Edition* **Note:** *AIX, OS/2, Windows NT and Windows 2000 platforms only.	• DB2 for MVS/ESA V3.1 and 4.1 • DB2 for OS/390 V5.1 • DB2 Universal Database for OS/390 V6.1 or later • DB2/400 V3.1 or later • DB2 Server for VM or VSE V5.1 or later

Table 1. Host and AS/400 multisite update scenarios that require SPM (continued)

TP Monitor Used?	Protocol	SPM Needed?	Product Required (choose One)	Host and AS/400 Database Supported
No	TCP/IP	No	• DB2 Connect Personal Edition • DB2 Connect Enterprise Edition • DB2 Universal Database Enterprise Edition • DB2 Universal Database Enterprise - Extended Edition	• DB2 for OS/390 V5.1 • DB2 Universal Database for OS/390 V6.1 or later
No	SNA	Yes	• DB2 Connect Enterprise Edition* • DB2 Universal Database Enterprise Edition* • DB2 Universal Database Enterprise - Extended Edition* **Note:** *AIX, OS/2, Windows NT and Windows 2000 platforms only.	• DB2 for MVS/ESA V3.1 and 4.1 • DB2 for OS/390 V5.1 • DB2 Universal Database for OS/390 V6.1 or later • DB2/400 V3.1 or later • DB2 Server for VM and VSE V5.1 or later

Note: A distributed transaction can update any mix of supported database servers. For example, your application can update several tables in DB2

UDB on Windows NT, a DB2 for OS/390 database and a DB2/400 database all within a single transaction.

For more information about two-phase commit, as well as instructions for setting up for several popular TP monitors, refer to the *Administration Guide*.

You can also access the DB2 Product and Service Technical Library on the World Wide Web:

1. Go to the following Web page:
   http://www.ibm.com/software/data/db2/library/
2. Select the **DB2 Universal Database** link.
3. Search for ″Technotes″ using the search keywords ″DDCS″, ″SPM″, ″MTS″, ″CICS″, and ″ENCINA″.

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 through DB2 Connect.

IBM provides application programmers with tools to generate SQL requests for Windows, OS/2, and several UNIX platforms. These tools are part of the DB2 Application Development Client. The DB2 Application Development Client supports several API types: embedded SQL, JDBC, SQLJ, and the DB2 Call Level Interface (DB2 CLI). These APIs can be used by programmers to build applications in a variety of programming languages. For more information about these APIs, refer to *Application Building Guide*.

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

Using the Control Center to Enable Multisite Updates

You can use the Control Center to provide multisite updates. The procedure is simple, and is outlined below. For more information about the multisite update configuration process, including how to configure your system manually, refer to the on-line *Connectivity Supplement*.

Starting the Multisite Update Wizard

From the Control Center, click the [+] sign to expand the tree view. With the right mouse button, select the instance that you wish to configure. A pop-up menu opens. Select **Multisite Update —> Configure** menu item.

Wizard Steps

The Wizard provides a notebook-type interface. Each page of the wizard will prompt you for certain information about your configuration. The pages are shown below in the order in which you will encounter them.

Step 1. Specify a Transaction Processor (TP) monitor.

This field will show the defaults for the TP monitor you have enabled. If you do not want to use a TP monitor, select **Do Not Use a TP Monitor**.

Step 2. Specify the communications protocols you will use.

Step 3. Specify a Transaction Manager database.

This panel defaults to the first database you connect to (1ST_CONN). You can leave this default or select another catalogued database.

Step 4. Specify the types of database servers involved in the update, and also whether or not TCP/IP is to be used exclusively.

Step 5. Specify the Syncpoint Manager settings.

This page will only appear if the settings on the previous page indicate that you need to use DB2's Syncpoint Manager in a multisite update scenario.

Testing the Multisite Update Feature

Step 1. Select the instance with the right mouse button and choose the **Multisite Update —> Test** menu option from the pop-up menu. The Test Multisite Update window opens.

Step 2. Select the databases you want to test from the available databases in the **Available databases** list box. You can use the arrow buttons in the middle to move selections to and from the **Selected databases** list box. You can also change the selected userid and password by directly editing them in the **Selected databases** list box.

Step 3. When you have finished your selection, click **OK** at the bottom of the window. The Multisite Update Test Result window opens.

Step 4. The Multisite Update Test Result window shows which of the databases you selected succeeded or failed the update test. The window will show SQL codes and error messages for those that failed.

Related DRDA Online Publications

The following online publications contain useful information related to DRDA.

For AS/400:
http://www.as400.ibm.com/db2/v4r4book.htm

For OS/390:
http://www.ibm.com/software/data/db2/os390/library.html

For DataJoiner:
http://www.ibm.com/software/data/datajoiner/library.html

For Database/Data Management online publications:
http://www.ibm.com/software/data/pubs/

Chapter 3. Scenarios for Using DB2 Connect

DB2 Connect can provide a variety of solutions to your host or AS/400 database access needs. This section will outline several scenarios that may apply to your particular needs or environment.

DB2 Connect Personal Edition is used to connect a single Windows 32–bit system, Linux, or OS/2 workstation to an S/390 or AS/400 database. DB2 Connect Personal Edition is best suited for environments where native TCP/IP support is provided by the database servers, and the application being deployed is a traditional 2-tier client-server application.

For example, DB2 Connect Personal Edition is a good choice for enabling traditional 2-tier VisualBasic and Microsoft Access applications. Applications that require mid-tier application server need to use DB2 Connect Enterprise Edition. For information on deployment scenarios using DB2 Connect Personal Edition, see "Direct Database Access" on page 22.

DB2 Connect Enterprise Edition is often installed on an intermediate server to connect DB2 clients to a host or AS/400 database. It can also be used on machines where multiple local users want to access the host or AS/400 servers directly.

For example, DB2 Connect Enterprise Edition may be installed on a large machine with many local users. It may also be installed on a web server, Transaction Processor (TP) monitor, or other 3-tier application servers with multiple local SQL application processes and threads. In these cases, you can install DB2 Connect Enterprise Edition on the same machine for simplicity, or on a separate machine to off-load CPU cycles.

DB2 Connect Enterprise Edition is most appropriate for environments where:

- Host and AS/400 database servers do not support native TCP/IP connectivity and direct connectivity from desktop workstations via SNA is not desirable. See "DB2 Connect Enterprise Edition as a Connectivity Server" on page 24.
- Web servers run web-based applications. See "DB2 Connect and Web Applications" on page 26.
- Web servers run web-based application using data-aware Java applets.
- A middle-tier application server is used. See "Using DB2 Connect with Application Servers" on page 31.

- TP monitors, such as CICS, Encina, Microsoft Transaction Server (MTS), Tuxedo, Component Broker, and MQSeries, are used. See "Using DB2 Connect with Transaction Processing Monitors" on page 35.

DB2 Connect Unlimited Edition is a unique package offering that allows complete flexibility of DB2 Connect deployment and simplifies product selection and licensing. This product contains both DB2 Connect Personal Edition and DB2 Connect Enterprise Edition with license terms and conditions that allow the unlimited deployment of any DB2 Connect product. License charges are based on the size of the System/390 that DB2 Connect users will be working with. This new package offering is only available for OS/390 systems and licensing is only valid for DB2 for OS/390 data sources.

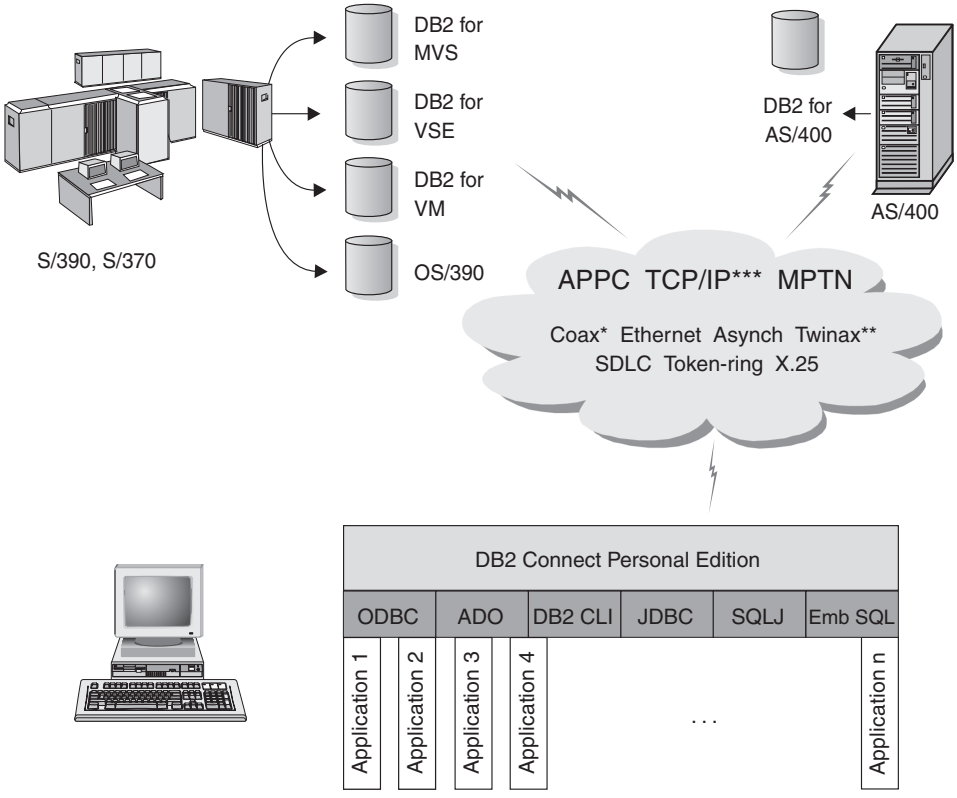
Direct Database Access

DB2 Connect's basic feature is providing a direct connection to a host database from desktop applications running on Windows 32–bit systems, Linux, or OS/2 workstations. DB2 Connect Personal Edition is the simplest way to implement this solution.

Each workstation that has DB2 Connect Personal Edition installed can establish a direct TCP/IP connection to DB2 for OS/390, DB2/400 and DB2 UDB for Windows NT, Windows 2000, UNIX and OS/2 servers. In addition, applications can connect to and update multiple DB2 family databases in the same transaction with the complete data integrity provided by the 2-phase commit protocol.

DB2 Connect Personal Edition has integrated APPC support as well, to communicate with DB2 for MVS and other host and AS/400 databases that require APPC. However, the use of TCP/IP is highly recommended instead of SNA, when native TCP/IP support is available.

Figure 3 on page 23 shows workstations directly connected to a host or AS/400 database server. Each workstation has DB2 Connect Personal Edition installed.



Not all protocols are supported for all platforms.

* For Host connections only

** For AS/400

*** TCP/IP connectivity requires DB2 for OS/390 V5R1, DB2 for AS/400 V4R2, or DB2 for VM V6.1

Figure 3. Direct Connection Between DB2 Connect and a host or AS/400 database server

Notes:

1. For information on what protocols are supported on which DRDA ARs and their respective host and AS/400 DRDA, refer to the appropriate *DB2 Connect Quick Beginnings* book.
2. 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.
3. The DB2 Application Development Client is now a part of the DB2 Connect package and can be installed if a customer wants to use it for

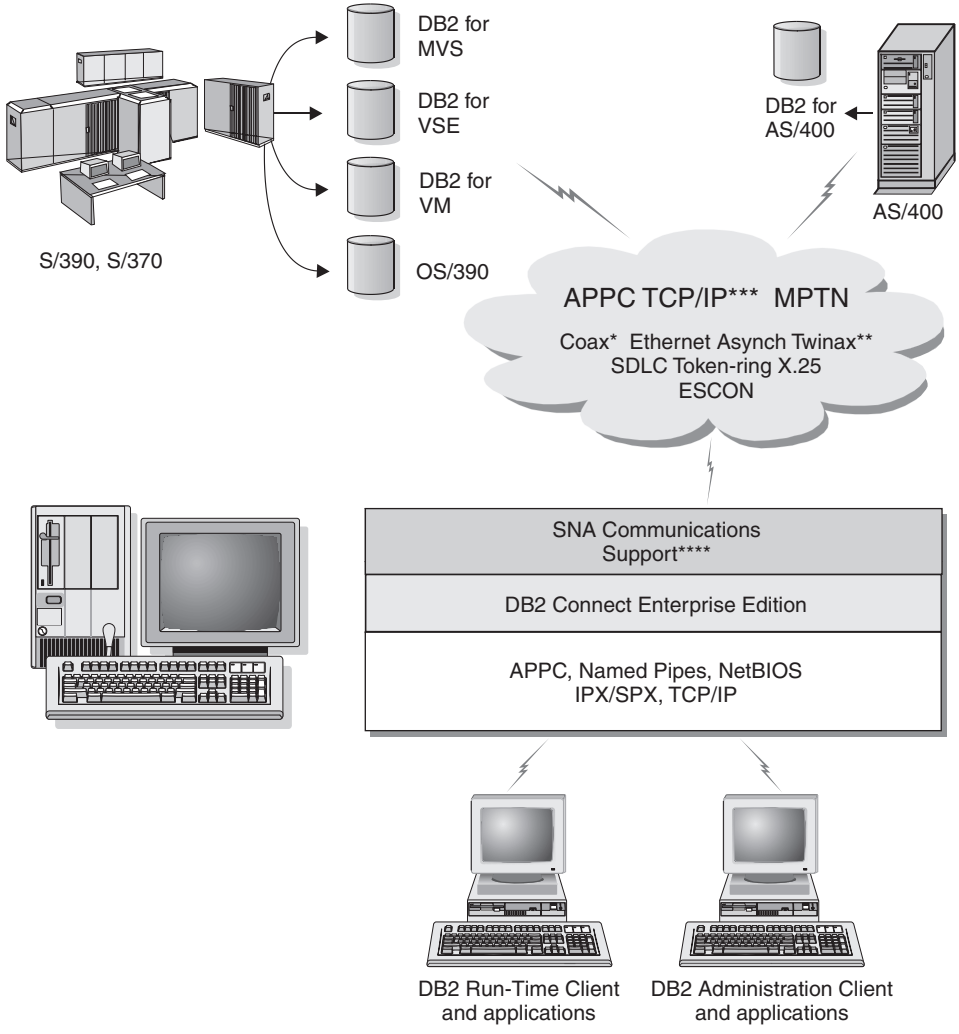
application development. In addition, DB2 Connect now includes Stored Procedure Builder that can be used to build, test, and deploy stored procedures for DB2 for OS/390.

4. C programmers developing Windows applications that use Microsoft ODBC, OLE DB, or ActiveX Data Objects (ADO) should use the *Microsoft Open Database Connectivity Software Development Kit*. Programmers who want to develop applications using the Java programming language can use any Java development environments, such as IBM's VisualAge for Java.

DB2 Connect Enterprise Edition as a Connectivity Server

A DB2 Connect server enables multiple clients to connect to host or AS/400 data and can significantly reduce the effort that is required to establish and maintain access to enterprise data. Figure 4 on page 25 illustrates IBM's solution for environments in which you want a DB2 client to make an indirect connection to a host or AS/400 database server through DB2 Connect Enterprise Edition.

In the following example, you could replace the DB2 Connect server with a DB2 UDB Enterprise Edition or Enterprise - Extended Edition server that has the DB2 Connect Server Support component installed.



Not all protocols are supported for all platforms.

* For Host connections only
** For AS/400
*** TCP/IP connectivity requires DB2 for OS/390 V5R1, DB2 for AS/400 V4R2, or DB2 for VM V6.1
**** SNA Comm Support is specific for each operating system and is required only in cases where native TCP/IP conectivity is not available.

Figure 4. DB2 Connect Enterprise Edition

DB2 Connect and Web Applications

The web browser is rapidly becoming a standard interface for everything from on-line catalogs to intranet applications. For simple web applications, a web server alone may be sufficient. For high-volume applications that may require database access and transaction processing, IBM offers solutions that use DB2 Connect to manage very high numbers of simultaneous transactions over the web.

This section describes web-based business solutions that can benefit from using DB2 Connect.

Advantages and Limitations of Traditional CGI Programming

e-business applications on the World Wide Web typically use the Common Gateway Interface (CGI) to enable users to query back-end databases. Many companies also use web applications internally, and these usually have a database in the background as well.

Users fill out forms on a web page, and these forms are submitted via CGI to applications or scripts on the web server. The script will in turn use a provided database API to submit SQL queries to a host database. The same script can then build a web (HTML) page with results of the query and send it back to be displayed by the user's web browser. For example, an on-line catalog, in which the user can query the availability and current price of particular goods or services.

CGI applications can be simple to design and easy to maintain. Since the CGI standard is both operating system- and language-independent, it is available on nearly all computing platforms. CGI programs can be written in C++, or in a scripting language such as Perl.

While CGI may seem like an ideal solution for web-based applications, it has significant shortcomings. The programming environment for CGI is not as sophisticated as other APIs. In addition, there is a scalability issue that will affect any large-scale e-commerce operation. Every time a CGI application is invoked, a new process is created on the web server. Each instance must make its own connection to the database, and each instance submits its own query. In high-volume transactional environments, this limitation can create significant performance issues.

You can use DB2 Connect with a web server to create robust, high-volume e-commerce applications. DB2 Connect provides several solutions that improve web-based application performance. Stored procedures (see "DB2 Connect on the Web Server" on page 27) allow DB2 Connect users to reduce the number of queries being sent to the database.

Connection pooling (see "Connection Pooling" on page 28) reduces the frequency of connections and disconnections to and from a database. For large operations where the limitations of CGI become important, see IBM Net.Data (see "Net.Data" on page 29) and WebSphere (see "IBM WebSphere" on page 30) provide non-CGI connections to large enterprise applications.

DB2 Connect on the Web Server

IBM provides HTTP (Web) servers with all DB2 Connect products for OS/2, UNIX, Windows NT, and Windows 2000. DB2 Connect Enterprise Edition provides out-of-the-box support for Apache or Lotus Domino Go web servers and can also work with any other web server such as Microsoft Internet Information Server or Netscape Enterprise Server.

If you are working with the DB2 family of databases running on OS/390, AS/400, VM, and VSE systems, DB2 Connect Enterprise Edition is required on the Web server. DB2 Connect Enterprise Edition will provide the libraries and communication interfaces to enable Web servers to access these host and AS/400 platforms. Either TCP/IP or SNA can be used to communicate between the Web server and a database running on OS/390, AS/400, VM or VSE.

Note: IBM web solutions provide the ability to work with multiple databases within the same CGI script or within the same transaction in a CGI script.

The next two sections discuss performance enhancements available to CGI applications that access DB2 databases. Later sections will explore alternatives to standard CGI such as Java.

Stored Procedures

An important consideration for web applications, as in the client/server world, is to minimize the traffic that occurs between the HTTP server and the back end database. This consideration is particularly important in high-volume transactional processing, which is the heart of most e-business applications.

The recommended approach is to combine CGI application programming with the programming and business logic encapsulated in stored procedures. DB2 Universal Database on OS/2, UNIX, and Windows, and DB2 on OS/390, AS/400 and VSE all share the same parameter convention for invoking stored procedures.

As with regular CGI, the web browser submits the form to the web server, where the CGI script is run. However, instead of each individual SQL statement being sent to the DB2 database, a request to execute a stored procedure is sent. This stored procedure encapsulates a number of SQL

statements that would have otherwise been run individually. Stored procedures reduce the number of messages flowing back and forth between the CGI script and the back end database.

The key benefit of stored procedures is reduced network traffic between the HTTP server and the DB2 database back end. For more information about stored procedures, refer to the *Application Development Guide* or the DB2 Stored Procedure Builder online help.

Connection Pooling

Establishing a connection from a DB2 Connect server to the host requires computing resources and time. In an environment where thousands of clients frequently connect to and disconnect from the host through the DB2 Connect server, a substantial portion of processing time is spent in establishing connections and dropping connections.

DB2 Connect's connection pooling provides a significant performance improvement in such environments. DB2 Connect maintains open connections to the database in an available pool. When a client requests a connection, it can be provided from this pool of ready connections. Connection pooling significantly reduces the overhead typically spent on opening and closing these connections.

For more information on how connection pooling works, see "Connection Pooling" on page 137.

DB2 Connect as a Java Application Server

Many of the shortcomings of CGI can be overcome by moving away from it and using Java instead. IBM provides both applets and applications that allow you to substitute Java for CGI at every stage of a web transaction. The solutions IBM provides allow for a mix of techniques, which means you can use scripting solutions such as Net.Data and Microsoft Active Server Pages with DB2, or move towards a more robust implementation provided by a Java application server such as IBM WebSphere.

There are two Application Programming Interfaces (APIs) for Java programmers. The first, JDBC, is supported for using Java to develop data-aware Java Applets, Java Applications as well as Java servlets, Java server pages (JSP) and Enterprise Java Beans (EJB). JDBC is a call-level or method invocation API. The other Java API is SQLJ. SQLJ provides the ability to specify SQL in-line within a Java program. DB2 can use both APIs, on either the client or server side of a web transaction.

On the client side, applets, data-aware applets, and applications are supported. On the database side Java enablement consists of database objects, such as user-defined functions and stored procedures.

For DB2 for OS/390, DB2 for VSE and VM, and DB2 for OS/400, there are two different ways to deploy a Java application. You can use the direct connectivity provided by DB2 Connect Personal Edition with TCP/IP or SNA, or you can choose to go through a DB2 Connect Enterprise Edition server that will provide connectivity to the mainframe or the AS/400 back end.

In both cases, the user on the Web does not require any special software to access the database, only a standard web browser. The only thing that needs to be installed is a DB2 Connect server and any industry standard Web server. If the web server and DB2 Connect are not on the same physical machines, a DB2 client needs to be installed on the web server.

For DB2 for OS/390 the key component is DB2 Connect Enterprise Edition running on a mid-tier server. This is the component that will provide JDBC server enablement, in addition to connecting to the DB2 for OS/390, VSE and VM, or AS/400 server. Again, there is no need for any special software for the client's web browser.

IBM provides an extensive set of tools for developing Java applications and applets. For database connectivity, DB2 Developer's Edition provides a complete kit containing VisualAge for Java Professional Edition, WebSphere Application Server, Net.Data, as well as DB2 Universal Database and DB2 Connect for testing. IBM VisualAge for Java Enterprise Edition also contains development tools for large-scale enterprise applications. Third-party tools such as Borland JBuilder or Symantec Visual Cafe will also work with IBM's database solutions.

Net.Data

Net.Data, part of the DB2 Universal Database and DB2 Connect family, is a set of application development tools designed to help you create and maintain web-based transaction applications. You can use Net.Data to access and alter data stored on a DB2 UDB for OS/2, Windows NT, Windows 2000, UNIX, OS/390, VM, VSE, and OS/400. The applications you create using Net.Data are stored on a web server and can be activated through a web browser.

Net.Data uses macros, or templates, to allow users with a basic understanding of HTML and SQL to build very sophisticated web applications. A macro is a text file that can be composed of Java, Java Scripts, HTML tags, and built-in functions. These macros can then be used to generate dynamic web pages with predefined layout, variables, and functions.

A basic Net.Data macro has seven distinct sections:

- Common sections, which basically serve as documentation aids for the programmer.
- Define section, which provides a place to specify variable definitions.

- Function section, which contains the main programming logic.
- Report section, which specifies the formatting logic for the Net.Data macro output.
- HTML section, which contains most of the HTML used in the web page.
- Include section, which is just a convenient way of including common parts of the macro that can be reused by other macros.
- Message section, where the error handling is provided.

The key feature of Net.Data, specifically for DB2, is that there is no client deployment required. The client in this implementation is simply a web browser.

The Net.Data processor is installed together with the DB2 Universal Database on a Windows NT, Windows 2000, OS/2, or UNIX workstation along with the Web server. When connecting to DB2 OS/390, DB2 for VSE and VM, and 400, all of the Net.Data infrastructure is deployed on a DB2 Connect server, along with a Web server.

IBM WebSphere

IBM WebSphere provides a more complete e-business solution than is possible with traditional CGI programming. WebSphere application servers not only perform the scripting possibilities of CGI, but also allow you to provide complex and high-end services through the web, using servlets, Active Server Pages, and enterprise JavaBeans. With WebSphere you can:

- Exploit industry standards to speed development and maximize inter-operability;
- Plug in third-party tools technologies and application frameworks;
- Analyze Web site content performance and usage;
- Scale your site easily to accommodate more users and maintain throughput;
- Deploy across a number of major operating environments (IBM AIX, HP-UX, Linux, Novell NetWare, IBM OS/2, IBM OS/390, IBM OS/400, Sun Solaris, Microsoft Windows NT and Windows 2000);
- Use your existing web server, including those from Apache, IBM, Netscape, and Microsoft.

WebSphere is not one product, but a family of three products addressing three different target markets. The heart of the WebSphere solution is the WebSphere application server.

The WebSphere application server provides the environment for three types of objects. One is Java server pages, which are analogous to Active Server Pages. The second component consists of Java servlets, and the third is enterprise JavaBeans. Enterprise JavaBeans are the emerging standard for deploying very large-scale, robust enterprise-class applications.

Additionally, Data Access JavaBeans provide very sophisticated database functions specifically tailored to DB2. DB2 is also directly accessible via JDBC and SQLJ. Both COM+ as well as CORBA are also supported.

WebSphere applications can be deployed on the same platform as the web server and DB2 Universal Database. In the case of the DB2 for OS/390, VM, VSE, and AS/400, WebSphere is deployed on the same platform as DB2 Connect Enterprise Edition.

There are several WebSphere solutions, as well as the Web Studio and WebSphere Performance Packs. The three WebSphere versions are:

Standard Edition
For Web site producers this server enables the use of Java servlets and JSP technology to quickly and easily transform Web sites and portals from static pages to vital sources of personalized dynamic Web content. It also includes industry-leading XML support for sharing information and data easily across groups or between enterprises and built-in site analysis technology that provides performance and usage information to help you maximize your company's return on its Web site investment.

Advanced Edition
For application programmers this high-performance EJB server enables deployment of business logic using EJB components. It provides scalability security connectivity and Java support and includes all of the functionality of the Standard Edition.

Enterprise Edition
For enterprise architects this server integrates disparate business systems across your organization to build robust e-business applications and maximize the reuse of resources. Enterprise Edition incorporates the capabilities of the award-wining IBM TXSeries- and Component Broker technologies. It also includes all of the functionality of the Advanced and Standard Editions.

Using DB2 Connect with Application Servers

The rise of client-server applications allowed application designers to improve usability and decrease training costs by providing applications with graphical user interfaces on platforms such as Windows and OS/2. At the same time, it allowed the flexibility of delegating database management function to robust database servers on a variety of operating systems and hardware platforms.

The client-server model, where application logic is distributed to client workstations, is commonly referred to as *2-tier client server*. In the 2-tier model, the application is deployed on the client tier and database server

implements the server or the back-end tier. As seen in "Direct Database Access" on page 22, DB2 Connect provides complete support for 2-tier client-server applications, where database servers are DB2 for OS/390, DB2 for MVS/ESA, DB2/400, or DB2 for VM and VSE.

With the increase in the size of the client-server applications, it became apparent that the 2-tier client-server model had significant limitations. Distributing large amounts of business logic to hundreds or even thousands of client workstations made change management a complex and costly undertaking. Any change in business rules required replacement of the client portion of the application. Often these application rollouts had to be on all client workstations in the enterprise at the same time to ensure that business rules are being applied consistently.

Another shortcoming of the 2-tier client-server model became apparent with scale is the amount of resources that are consumed by such applications. Deploying hundreds or thousands of *fat clients*, as 2-tier clients are often called, increased demands on processing power and capacity of each client workstation. Moreover, the demands on the database server are also greatly increased as each client required a dedicated database connection and the resources associated with maintaining such a connection. While the 2-tier client-server dependency of distributing business logic can be somewhat reduced by extensive use of stored procedures, the other shortcomings are not easily addressed without changes to the model.

An Application Server Solution

As the cost and complexity of 2-tier client-server applications escalated, most of the largest applications embarked on the path to multi-tier client-server. Under the multi-tier model, the role of the database tier remains unchanged. However, the client tier is supplemented by a one or more middle tiers; typically one, therefore the name *3-tier*.

In the 3-tier model, the client is relegated to handling user interactions and does not contain any business logic. The middle-tier is comprised of one or more application servers. The goal of the application server is to provide robust, cost-efficient implementation of the logic behind the business processes and business rules. As with the 2-tier model, the business rules implementation is often supplemented by using stored procedures to improve performance.

Because client workstations no longer implement the bulk of the application logic and are only handling user interactions, the resource requirements for the client tier are greatly reduced. As a matter of fact, the client tier in the 3-tier model is often called *thin client*. In addition, because a centralized application server is handling requests from all of the clients, it has the ability

to share resources, such as database connections between all of the clients. As a result, the database server no longer has to maintain dedicated connections for each application user.

Many examples of 3-tier applications servers exist in the industry today. Almost all Enterprise Resource Planning (ERP) vendors implement their applications using the 3-tier model, such as SAP R/3 and PeopleSoft V7 applications. Other examples include leading Enterprise Relationship Management vendors, such as Siebel and Vantive.

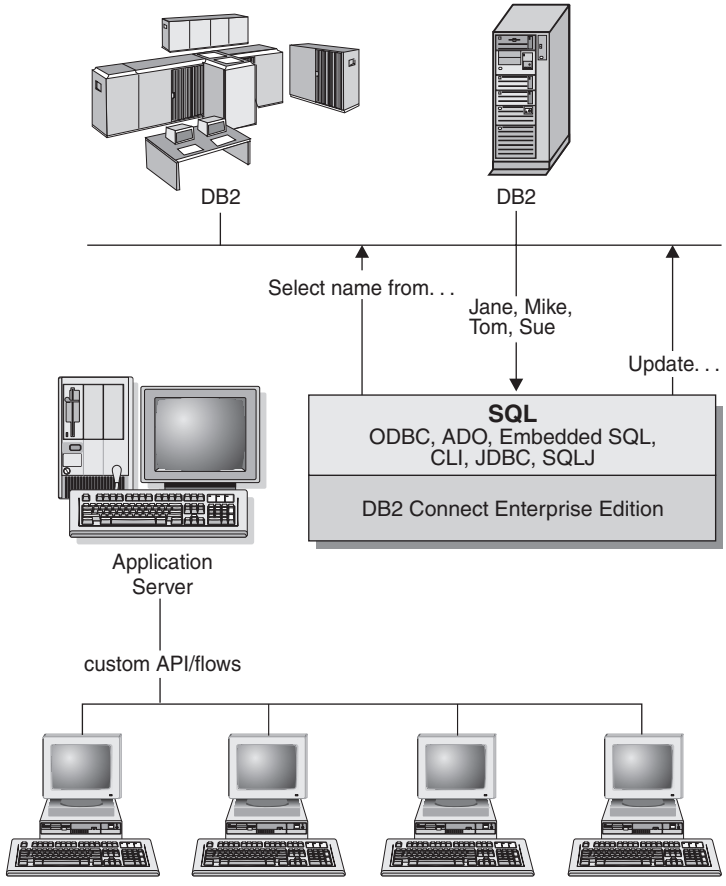
Application Servers and DB2 Connect

DB2 Connect Enterprise Edition servers provide comprehensive support for deploying multi-tier applications. The support provided by DB2 Connect includes a variety of APIs that can be used to develop application logic (ODBC, ADO, DB2 CLI, Embedded SQL, JDBC, and SQLJ), as well as a complete communication infrastructure for interacting with DB2 Family database servers.

DB2 Connect also supports implementations in which a database tier is comprised of multiple DB2 Family database servers. This allows application servers to implement transactions that update data residing on multiple database servers in a single transaction.

The integrity of such distributed transactions is assured by the 2-phase commit protocol support provided by DB2 Connect. For example, an application can update data in a DB2 for OS/390 database and DB2 UDB on Windows NT in the same transaction. If distributed request support is installed and enabled, the application can read an Oracle database and update a DB2 family database in the same transaction.

In the following diagram, the APIs as well as the connectivity mechanism between the application server and the back-end database servers is provided by DB2 Connect Enterprise Edition.



Advanced features of DB2 Connect, such as connection pooling (see "Connection Pooling" on page 137) and connection concentrator (see "DB2 Connect Connection Concentrator" on page 139), greatly reduce application resource requirements and simplify application server implementation.

DB2 Connect and Application Server Configurations

DB2 Connect Enterprise Edition product (available on its own or as part of the DB2 Connect Unlimited Edition product package) is required for use with application servers. DB2 Connect Personal Edition is not supported and is not licensed for use with application servers. In addition, customers implementing application servers should review terms and conditions provided with their copy of DB2 Connect to understand the number of user licenses that need to be acquired. There are two deployment methods for DB2 Connect in the application server environment. DB2 Connect Enterprise Edition installed on:

- The application server machine; or
- A separate communication server machine.

In most situations, installing a copy of DB2 Connect on the same server as the application server itself is the preferred solution. Installing DB2 Connect on the application server allows it to participate in any fail-over and load-balancing scheme that an application server may be implementing. This setup can potentially provide better performance since it eliminates an additional network hop that is required when DB2 Connect is installed on a separate server. Furthermore, the administration can be simplified since there is no need for an installing and maintaining an additional server.

Installing DB2 Connect on a separate server is a good option in situations where DB2 Connect Enterprise Edition is not available for the operating system or hardware platform where application server is running. For example, if the application server is deployed on a Silicone Graphics (SGI) or SCO UnixWare server, deploying DB2 Connect on a separate server is the only option as DB2 Connect Enterprise Edition is not available on these platforms.

Using DB2 Connect with Transaction Processing Monitors

In the previous section you learned about using DB2 Connect with an application server. An application server permits a large number of users to execute applications using a minimum of system resources.

An application server can be extended to allow coordinated transactions to be invoked from applications executed by the application server. This transaction coordination is generally known as a Transaction Processing (TP) monitor. A TP monitor works in conjunction with an application server.

A *transaction* can be thought of as a routine event, usually a request for service, in running the day-to-day operations of an organization. The orderly processing of transactions is the type of work for which TP monitors were designed.

Every organization has rules and procedures that describe how it is supposed to operate. The user applications which implement these rules can be called *business logic*. The transactions these business applications execute are often referred to as Transaction Processing or Online Transaction Processing (OLTP).

The key characteristics of commercial OLTP are:

Many Users
: It is common for transaction processing to be used by the majority of the people in an organization, since so many people affect the current state of the business.

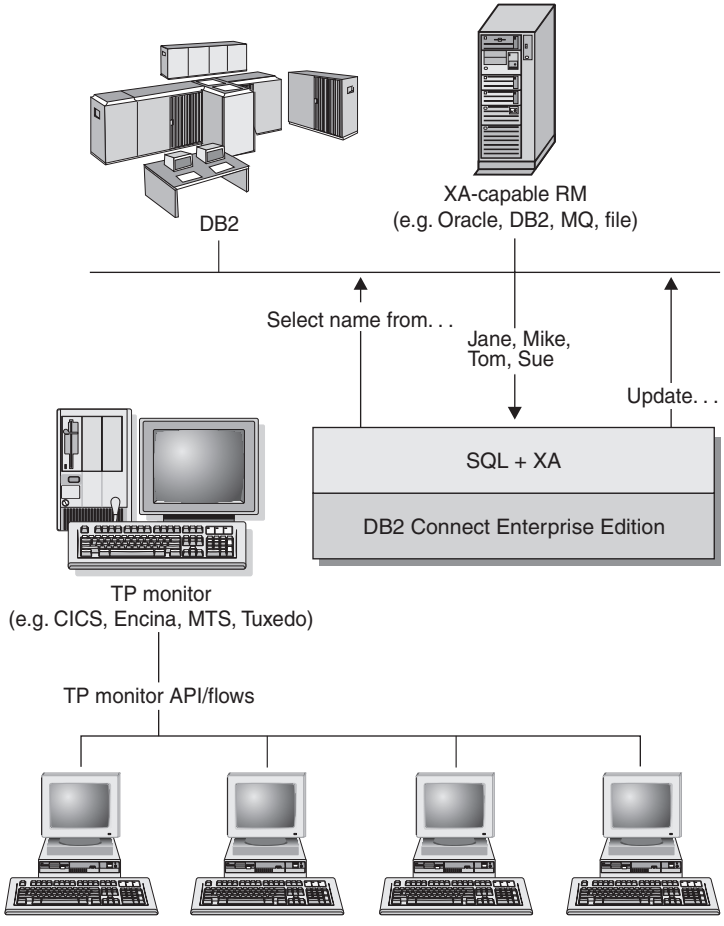
Repetitive
Most interactions with the computer tend to be the same process executed over and over again. For example, entering an order or processing payments are used many times every day.

Short Interactions
Most interactions that people in the organization have with the transaction processing system are short in duration.

Shared Data
Since data represents the state of the organization, there can only be a single copy of the data.

Data Integrity
The data must represent the current state of the organization, and must be internally consistent. For example, every order must be associated with a customer record.

Low Cost/Transaction
Since the transaction processing represents a direct cost of doing business, the cost of the system must be a minimum. DB2 Connect allows applications under the control of an application server running on UNIX, Windows NT, Windows 2000 or OS/2 to execute transactions against remote LAN, host, and AS/400 database servers and have these transactions coordinated by a TP monitor.



In this figure, the APIs, as well as the connectivity mechanism between the application server and the back-end database servers, are provided by DB2 Connect Enterprise Edition.

Examples of TP Monitors

The most common TP monitors on the market today are:

- IBM TxSeries CICS
- IBM TxSeries Encina Monitor
- BEA Tuxedo.

Microsoft Transaction Server Remote S/390, AS/400, and LAN database servers can be use within transactions coordinated by these TP monitors.

Tuxedo and DB2 Connect

With DB2 Connect Version 6 and earlier versions, Tuxedo based applications were limited to read only access to host and AS/400 database servers. This

restriction has been removed with DB2 Connect Version 7. Tuxedo based applications can now update host and AS/400 database servers within a Tuxedo coordinated transaction. Special configuration requirements and restrictions apply. For more information, see "DB2 Connect Connection Concentrator" on page 139.

X/Open Distributed Transaction Processing (DTP) Model

An application executing business logic may be required to update multiple resources within a single transaction. For example, a bank application which implements a transfer of money from one account to another could require debiting one database (the "from" account) and depositing to another database (the "to" account).

It is also possible that different vendors provide these two databases. For example, one database is a DB2 Universal Database for OS/390 and the other is an Oracle database. Rather than have every TP monitor implement each database vendor's proprietary transaction interface, a common transaction interface between a TP monitor and any resource accessed by an application has been defined. This interface is known as the *XA Interface*. A TP monitor that uses the XA Interface is referred to as an *XA compliant Transaction Manager (TM)*. An updatable resource that implements the XA interface is referred to as an *XA compliant Resource Manager (RM)*.

The above listed TP monitors are all XA compliant TMs. Remote host, AS/400, and DB2 UDB LAN-based database servers, when accessed via DB2 Connect, are XA compliant RMs. Therefore, any TP monitor which has an XA compliant TM can use host, AS/400, and LAN based DB2 UDB database servers within business applications executing transactions.

How to Use DB2 Connect With an XA Compliant Transaction Manager

This section describes the configuration steps necessary to use S/390 and AS/400 database servers within your TP monitor. This section assumes that you have an operational TP monitor and have installed DB2 Connect, as well as have configured and tested a connection to the host or AS/400 database server. For more detailed information, refer to *DB2 Connect Quick Beginnings* book.

The steps required to configure the most popular TP monitors are provided in the *Administration Guide*. There is no distinguishing between configuring for access to a LAN-based DB2 UDB database server versus a host or AS/400 database server. The following instructions outline the general configuration steps for TP monitors not listed in the *Administration Guide*.

To configure DB2 Connect to use S/390 and AS/400 database servers within your TP monitor, perform the following steps:

1. Configure the TP monitor so that it can access the DB2 XA Switch. The DB2 XA Switch provides the TP monitor with the addresses of DB2 Connect's XA APIs. Every TP monitor has a different way to do this. For information on providing the DB2 XA Switch to a TP monitor, refer to the *Administration Guide*.
2. Configure the TP monitor with DB2's XA_OPEN string. Each TP monitor has its own way to do this. For information on DB2 Connect's XA OPEN string, refer to the *Administration Guide*. For information on how to configure DB2's XA OPEN string for use by the TP monitor, refer to your TP monitor's documentation.
3. If required, modify the DB2 Connect Sync Point Manager (SPM) default configuration parameters. Host and AS/400 database servers do not yet support the XA interface.

   The SPM is a component of DB2 Connect which maps the XA two phase commit protocol into the two phase commit protocol used by host and AS/400 database servers. By default, the DB2 instance has pre-defined values for the SPM configuration parameters. The most significant parameter is the database manager configuration parameter SPM_NAME. It defaults to a variant of the first seven characters of the TCP/IP hostname.

   If you are using TCP/IP to connect to DB2 for OS/390, then you should not have to change any of the default settings. In this case, there is no SPM configuration required since it is already operational. If you are using SNA to access host or AS/400 database servers, then you have to ensure the SPM_NAME value represents a valid SNA LU in your network. If the default SPM_NAME value is not acceptable then you should use the Multisite Update wizard to modify this value.

Chapter 4. Programming in a DB2 Connect Environment

This section provides some information about creating applications that use DB2 Connect. For more information, refer to the *CLI Guide and Reference*, the *Command Reference*, and the *Application Development Guide*.

Programming in a Distributed Environment

DB2 Connect lets an application program access data in DB2 databases on System/390 and AS/400 servers. For example, an application running on Windows can access data in a DB2 Universal Database for OS/390 database. You can create new applications, or modify existing applications to run in a host or AS/400 environment. You can also develop applications in one environment and port them to another.

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

- Embedded SQL, both static and dynamic
- The DB2 Call Level Interface
- The Microsoft ODBC API
- JDBC.

Some SQL statements differ among relational database products. You may encounter SQL statements that are:

- The same for all the database products that you use regardless of standards
- Documented in the *SQL Reference* and are therefore available in all IBM relational database products
- 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 host or AS/400 server. For information on limits on different platforms, such as the maximum column length, refer to 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.

When you program in a host or AS/400 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 host or AS/400 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 host or AS/400 server when you precompile to a host or AS/400 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 Universal Database 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.

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 Universal Database 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 host or AS/400 server may support only a subset of scalar functions for long fields; for example, DB2 Universal Database for OS/390 allows only the **LENGTH** and **SUBSTR** functions for long fields. Also, a host or AS/400 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 supported by DB2 Connect.

User Defined Types (UDTs)

Only User Defined Distinct Types are supported by DB2 Connect. Abstract Data Types are not.

ROWID Data Type

The ROWID data type is handled by DB2 Connect as VARCHAR for bit data.

64-bit Integer (BIGINT) data type

Eight byte (64-bit) integers are supported by DB2 Connect. The BIGINT internal data type is used to provide support for the cardinality of very large databases, while retaining data precision.

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 Universal Database for OS/390

ARI DB2 for VSE & VM

QSQ DB2 Universal Database 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).

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 host or AS/400 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 host or AS/400 server precompilers in the following ways:

- It makes only one pass through an application.
- 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 DB2 Connect support 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 host or AS/400 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 *Administrative 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 host or AS/400 server system has limitations on the use of these attributes:

DB2 Universal Database for OS/390
All four attributes can be different. The use of a different qualifier requires special administrative privileges. For more information on the conditions concerning the usage of these attributes, refer to the *Command Reference* for DB2 Universal Database for OS/390.

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 Universal Database 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
All four attributes can be different. The use of a different owner requires administrative authority and the binder must have CREATEIN privilege on the schema (if it already exists).

Note: DB2 Connect provides support for the *SET CURRENT PACKAGESET* command for DB2 Universal Database 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.

For a description of how null-terminated strings are handled when prepared with the LANGLEVEL option set to MIA or SAA1, refer to *Application Development Guide*.

By default, CNULREQD is set to YES. This causes null-terminated strings to be interpreted according to MIA standards. If connecting to a DB2 Universal Database for OS/390 server it is strongly recommended to set CNULREQD to YES. 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, refer to the *Application Development Guide*, the *Administrative API Reference* and the *Command Reference*.

Note: Database tables can now be stored on DB2 Universal Database for OS/390 in ASCII format. This permits faster exchange of data between DB2 Connect and DB2 Universal Database 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 Universal Database 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 Universal Database 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 unique constraint, including a primary 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 Universal Database 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 host or AS/400 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 database server that does not have SQLCODE mapping (a non-IBM database server). You can also turn off SQLCODE mapping.

  For more information, see “Chapter 11. SQLCODE Mapping” on page 121.

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.

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 Universal Database 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 Universal Database 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 host or AS/400 server does not support the isolation level that you specify, the next higher supported level is used.

Table 2 shows the result of each isolation level on each host or AS/400 application server.

Table 2. Isolation Levels

DB2 Connect	DB2 Universal Database for OS/390	DB2 for VSE & VM	DB2 Universal Database 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 Universal Database for AS/400 that matches RR. DB2 Universal Database 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 Universal Database 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.

  OS/390
  The schema name must be no more than 8 bytes long, the procedure name must be no more than 18 bytes long, and the stored procedure must be defined in the SYSIBM.SYSPROCEDURES catalog on the server.

  VSE or VM
  The procedure name must not be more than 18 bytes long and must be defined in the SYSTEM.SYSROUTINES catalog on the server.

  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, refer to the *SQL Reference*. For information on how to use stored procedures when writing application programs, refer to the *Application Development Guide*.

  You can invoke the server program on DB2 Universal Database with the same parameter convention that server programs use on DB2 Universal Database for OS/390, DB2 Universal Database for AS/400, or DB2 for VSE & VM. For more information on invoking DB2 Universal Database stored procedures, refer to the *Application Development Guide*. For more information on the parameter convention on other platforms, 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 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 Universal Database for OS/390 or DB2 Universal Database for AS/400 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.
- DB2 for VSE & VM Version 7 or higher and DB2 Universal Database for OS/390 Version 5.1 or higher are the only host or AS/400 Application Servers that can return the result sets of stored procedures at this time.

Stored Procedure Builder

DB2 Stored Procedure Builder provides an easy-to-use development environment for creating, installing, and testing stored procedures. It allows you to focus on creating your stored procedure logic rather than the details of registering, building, and installing stored procedures on a DB2 server. Additionally, with Stored Procedure Builder, you can develop stored procedures on one operating system and build them on other server operating systems.

Stored Procedure Builder is a graphical application that supports rapid development. Using Stored Procedure Builder, you can perform the following tasks:

- Create new stored procedures.
- Build stored procedures on local and remote DB2 servers.
- Modify and rebuild existing stored procedures.
- Test and debug the execution of installed stored procedures.

You can launch Stored Procedure Builder as a separate application from the DB2 Universal Database program group, or you can launch Stored Procedure Builder from any of the following development applications:

- Microsoft Visual Studio
- Microsoft Visual Basic
- IBM VisualAge for Java

You can also launch Stored Procedure Builder from Control Center for DB2 for OS/390. You can start Stored Procedure Builder as a separate process from the Control Center Tools menu, toolbar, or Stored Procedures folder. In addition, from the Stored Procedure Builder Project window, you can export one or more selected SQL stored procedures built to a DB2 for OS/390 server to a specified file capable of running within the Command Line Processor (CLP).

Stored Procedure Builder manages your work by using projects. Each Stored Procedure Builder project saves your connections to specific databases, such as

DB2 for OS/390 servers. In addition, you can create filters to display subsets of the stored procedures on each database. When opening a new or existing Stored Procedure Builder project, you can filter stored procedures so that you view stored procedures based on their name, schema, language, or collection ID (for OS/390 only).

Connection information is saved in a Stored Procedure Builder project; therefore, when you open an existing project, you are automatically prompted to enter your user id and password for the database. Using the Inserting SQL Stored Procedure wizard, you can build SQL stored procedures on a DB2 for OS/390 server. For a SQL stored procedure built to a DB2 for OS/390 server, you can set specific compile, pre-link, link, bind, runtime, WLM environment, and external security options.

Additionally, you can obtain SQL costing information about the SQL stored procedure, including information about CPU time and other DB2 costing information for the thread on which the SQL stored procedure is running. In particular, you can obtain costing information about latch/lock contention wait time, the number of getpages, the number of read I/Os, and the number of write I/Os.

To obtain costing information, Stored Procedure Builder connects to a DB2 for OS/390 server, executes the SQL statement, and calls a stored procedure (DSNWSPM) to find out how much CPU time the SQL stored procedure used.

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 host or AS/400 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, refer to the *SQL Reference*.

Multisite Update with DB2 Connect

DB2 Connect allows you to perform a multisite update, also known as two-phase commit. A multisite update is an update of 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.
- If you have SNA network connections, you can use two-phase commit support provided by the sync point manager function of DB2 Connect Enterprise Edition Version 7 on AIX, OS/2, and Windows NT. This enables the following host database servers to participate in a distributed unit of work:
  - DB2 for AS/400 Version 3.1 or later
  - DB2 for MVS/ESA Version 3.1 or later
  - DB2 for OS/390 Version 5.1 or later
  - DB2 for VM & VSE Version V5.1 or later.

  The above is true for native DB2 UDB applications and applications coordinated by an external Transaction Processing (TP) Monitor such as IBM TXSeries, CICS for Open Systems, BEA Tuxedo, Encina Monitor, and Microsoft Transaction Server.

  Note: For more information on BEA Tuxedo, see "Using DB2 Connect with Transaction Processing Monitors" on page 35.For more information on the XA concentrator, see "DB2 Connect Connection Concentrator" on page 139.
- If you have TCP/IP network connections, then a DB2 for OS/390 V5.1 or later server can participate in a distributed unit of work. If the application is controlled by a Transaction Processing Monitor such as IBM TXSeries, CICS for Open Systems, Encina Monitor, or Microsoft Transaction Server, then you must use the sync point manager.

  If a common DB2 Connect Enterprise Edition server is used by both native DB2 applications and TP monitor applications to access host data over TCP/IP connections then the sync point manager must be used.

  If a single DB2 Connect Enterprise Edition server is used to access host data using both SNA and TCP/IP network protocols and two phase commit is required, then the sync point manager must be used. This is true for both DB2 applications and TP monitor applications.

Host or AS/400 Server SQL Statements Supported by DB2 Connect

The following statements compile successfully for host or AS/400 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 host or AS/400 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.

Host or AS/400 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 Charge-Back Accounting on DB2 Universal Database for OS/390

Many DB2 Universal Database for OS/390 installations implement resource monitoring practices that 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 DB2 for database 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 only supported on MVS and OS/390 systems. The string is stored as an accounting record.

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 3 shows the system-generated fields. Each of these fields is padded to the right with blanks.

Table 3. 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 software. For example, the product ID for DB2 Universal Database Version 7 is `SQL07010`.
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 host or AS/400 database 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'SQL07010OS/2              cheque              SMITH   x'05'DEPT1

x'37'SQL07010OS/2              cheque              SMITH   x'00'
```

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

Sending Accounting Information to a DB2 for OS/390 Server

Many mainframe customers consider detailed accounting for resources that are used by different applications to be an important part of their operational procedures. DB2 for OS/390 provides extensive facilities for producing accounting reports that allow information systems departments to charge individual user departments for the mainframe resources they use. This

process is often called *chargeback accounting*. DB2 Connect products allow for accurate accounting for the host resources used by the PC and UNIX applications using existing accounting reports and procedures.

DB2 Connect implements this feature in a flexible way by providing:

- A default accounting string for all usage generated by a particular DB2 Connect Enterprise Edition server.
- Two mechanisms for individual users or applications to specify the account to which their usage should be charged.

Setting the Accounting String

The default accounting string is set by the DB2 Connect workstation's dft_account_str configuration parameter. This default mechanism is useful for database clients that do not have the capability to forward an accounting string to DB2 Connect. For example, applications developed prior to Version 2 products.

There are two ways for client applications to override the default accounting string set at the DB2 Connect server:

- Using the Set Accounting String API: sqlesact()

  The sqlesact() API is called before the application connects to a database. You should use this method because:

  – Calling an API does not incur the cost of reading a registry value.
  – You do not need to call this API again unless you want to use a new accounting string for future connect requests. If you are using the registry value, you need to end the application process, redefine DB2ACCOUNT, and then restart the process.

  For more information on using this API, refer to the *Administrative API Reference*.
- Using the DB2ACCOUNT registry value at the client workstation.

  If the sqlesact() API is not called prior to the first database connect request, the DB2ACCOUNT registry value is read. The accounting string is used for all subsequent database connect requests.

Note: When defining the accounting string, you should observe the following rules:

1. Use the characters A-Z, 0-9, or '_' (underscore).
2. Limit the accounting string to 199 bytes—longer strings are truncated.

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 Universal Database for OS/390 Reference for Remote DRDA Requesters and Servers* provides the latest information about chargeback accounting for DB2 Universal Database for OS/390 users.
- *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 Online Publications" on page 19.

Chapter 5. Running Your Own Applications

Various types of applications can access DB2 databases:

- Applications developed using the DB2 Application Development Client that include embedded SQL, APIs, stored procedures, user-defined functions or calls to the DB2 CLI.
- ODBC applications such as Lotus Approach.
- JDBC applications and applets.
- Net.Data macros containing HTML and SQL.

An application on a DB2 client can access a remote database without knowing its physical location. The DB2 client determines the location of the database, manages the transmission of the requests to the database server, and returns the results.

In general, to run a database client application, use the following steps:

Step 1. Ensure the server is configured and running.

Ensure that the database manager is started on the database server to which the application program is connecting. If it is not, you must issue the **db2start** command at the server before starting the application.

Step 2. Ensure that you can connect to the database that the application uses.

Step 3. Bind the utilities and the applications to the database. See "Binding Database Utilities" for information about binding the utilities.

Step 4. Run the application program.

Binding Database Utilities

You must bind the database utilities (import, export, reorg, the command line processor) and DB2 CLI bind files to each database before they can be used with that database. In a network environment, if you are using multiple clients that run on different operating systems or are at different versions or service levels of DB2, you must bind the utilities once for each operating system and DB2-version combination.

Binding a utility creates a *package*, which is an object that includes all of the information that is needed to process specific SQL statements from a single source file.

The bind files are grouped together in different .lst files in the bnd directory, under the installation directory. Each file is specific to a server.

Running CLI/ODBC Programs

The DB2 Call Level Interface (CLI) run-time environment and the DB2 CLI/ODBC driver are included with DB2 clients as optional components during install.

This support enables applications developed using ODBC and DB2 CLI APIs to work with any DB2 server. DB2 CLI application development support is provided by the DB2 Application Development Client which is packaged with your DB2 server.

Before DB2 CLI or ODBC applications can access DB2, the DB2 CLI packages must be bound on the server. Although this will occur automatically on the first connection if the user has the required authority to bind the packages, it is recommended that the administrator do this first with each version of the client on each platform that will access the server.

The following general steps are required on the client system to give DB2 CLI and ODBC applications access to DB2 databases. These instructions assume that you have successfully connected to DB2 using a valid user ID and password. Depending on the platform many of these steps are automatic. For complete details, see the section that deals specifically with your platform.

Step 1. Use the Client Configuration Assistant (CCA) to add the database (if you have separate client and server machines) so that its instances and databases can be made known to the Control Center, then add the instances and databases for that system. If you do not have access to this program you can use the **catalog** command in the command line processor.

Step 2. The DB2 CLI/ODBC driver is an optional component during the DB2 client install on Windows platforms. Be sure it is selected at that point. On OS/2 you must use the **Install ODBC Driver** icon to install both the DB2 CLI/ODBC driver and the ODBC driver manager. On UNIX platforms the DB2 CLI/ODBC driver is automatically installed with the client.

Step 3. To access the DB2 database from ODBC:

a. The ODBC Driver Manager (From Microsoft or other vendor) must already be installed (this is done by default during the installation of DB2 only on 32-bit Windows systems).

b. The DB2 databases must be registered as ODBC data sources. The ODBC driver manager does not read the DB2 catalog information; instead it references its own list of data sources.

c. If a DB2 table does not have a unique index then many ODBC applications will open it as read-only. A unique index should be created for each DB2 table that is to be updated by an ODBC application. Refer to the **CREATE INDEX** statement in the *SQL Reference*. Using the Control Center you would alter the settings of the table, then select the **Primary Key** tab and move one or more columns from the available columns list over to the primary key columns list. Any column you select as part of the primary key must be defined as NOT NULL.

Step 4. If necessary, you can set various CLI/ODBC Configuration Keywords to modify the behavior of DB2 CLI/ODBC and the applications using it.

If you followed the above steps to install ODBC support, and added DB2 databases as ODBC data sources, your ODBC applications will now be able to access them.

Platform Specific Details for CLI/ODBC Access

The platform specific details on how to give DB2 CLI and ODBC applications access to DB2 are divided into the following categories:

- "Windows 32-bit operating systems Client Access to DB2 using CLI/ODBC"
- "OS/2 Client Access to DB2 using CLI/ODBC" on page 63

Windows 32-bit operating systems Client Access to DB2 using CLI/ODBC

Before DB2 CLI and ODBC applications can successfully access a DB2 database from a Windows client, perform the following steps on the client system:

Step 1. The DB2 database (and node if the database is remote) must be cataloged. To do so, use the CCA (or the command line processor).

For more information refer to the online help in the CCA (or the **CATALOG DATABASE** and **CATALOG NODE** commands in the *Command Reference*).

Step 2. Verify that the Microsoft ODBC Driver Manager and the DB2 CLI/ODBC driver are installed. On Windows 32-bit operating systems they are both installed with DB2 unless the ODBC component is manually unselected during the install. DB2 will not overwite a newer version of the Microsoft ODBC Driver Manager if one is found.

To verify that they both exist on the machine:

a. Start the Microsoft ODBC Data Sources icon in the Control Panel, or run the **odbcad32.exe** command from the command line.

b. Click on the **Drivers** tab.

c. Verify that ″IBM DB2 ODBC DRIVER″ is shown in the list.

If either the Microsoft ODBC Driver Manager or the IBM DB2 CLI/ODBC driver is not installed, then rerun the DB2 install and select the ODBC component on Windows 32-bit operating systems.

Step 3. Register the DB2 database with the ODBC driver manager as a *data source*. On Windows 32-bit operating systems you can make the data source available to all users of the system (a system data source), or only the current user (a user data source). Use either of these methods to add the data source:

- Using the CCA:
  a. Select the DB2 database alias that you want to add as a data source.
  b. Click on the **Properties** push button. The Database Properties window opens.
  c. Select the **Register this database for ODBC** check box.
  d. On Windows 32-bit operating systems you can use the radio buttons to add the data source as either a user or system data source.
- Using the **Microsoft 32-bit ODBC Administration tool**, which you can access from the icon in the Control Panel or by running **odbcad32.exe** from the command line:
  a. On Windows 32-bit operating systems the list of user data sources appears by default. If you want to add a system data source click on the **System DSN** button, or the **System DSN** tab (depending on the platform).
  b. Click on the **Add** push button.
  c. Double-click on the IBM DB2 ODBC Driver in the list.
  d. Select the DB2 database to add and click on **OK**.
- On Windows 32-bit operating systems there is a command that can be issued in the command line processor to register the DB2 database with the ODBC driver manager as a data source. An administrator could create a command line processor script to register the required databases. This script could then be run on all of the machines that require access to the DB2 databases through ODBC.

  The *Command Reference* contains more information on the CATALOG command:

  ```
  CATALOG [ user | system ] ODBC DATA SOURCE
  ```

Step 4. Configure the DB2 CLI/ODBC driver using the CCA: (Optional)

a. Select the DB2 database alias you want to configure.

b. Click on the **Properties** push button. The Database Properties window opens.

c. Click on the **Settings** push button. The CLI/ODBC Settings window opens.

d. Click on the **Advanced** push button. You can set the configuration keywords in the window that opens. These keywords are associated with the database *alias name*, and affect all DB2 CLI/ODBC applications that access the database. The online help explains all of the keywords, as does and the *Installation and Configuration Supplement* online manual.

Step 5. If you have installed ODBC access (as described above), you can now access DB2 data using ODBC applications. Start the ODBC application and go to the Open window. Select the **ODBC databases** file type. The DB2 databases that you added as ODBC data sources will be selectable from the list. Many ODBC applications will open the table as read-only unless a unique index exists.

OS/2 Client Access to DB2 using CLI/ODBC

Before DB2 CLI and ODBC applications can successfully access a DB2 database from an OS/2 client, perform the following steps on the client system:

1. The DB2 database (and node if the database is remote) must be cataloged. To do so, use the CCA (or the command line processor).

   For more information, see the online help in the CCA. (or the **CATALOG DATABASE** and **CATALOG NODE** commands in the *Command Reference*).

2. If you are using ODBC applications to access DB2 data, perform the following steps. (If you are using only CLI applications, skip this step and go to the next step.)

   a. Check that there is an ODBC Driver Manager installed. The ODBC Driver Manager is not installed with DB2; we suggest you use the Driver Manager that was shipped with your ODBC application. Also ensure that the DB2 CLI/ODBC driver is installed:

      1) Run the ODBC Administration tool as described in its documentation. This is usually done in one of two ways:

         - Double-click on the **ODBC** Folder in OS/2, and double-click on the **ODBC Administrator** icon.
         - Run **odbcadm.exe** from the command line.

         The Data Sources window opens.

      2) Click on the **Drivers** push button. The Drivers window opens.

      3) Verify that ″IBM DB2 ODBC DRIVER″ is shown in the list.

   If the ODBC Driver Manager is not installed then follow the installation instructions that came with your ODBC application. If the

IBM DB2 CLI/ODBC driver is not installed then double-click on the **Install ODBC Driver** icon in the DB2 folder to install the DB2 CLI/ODBC driver.

b. Register the DB2 database with the ODBC driver manager as a *data source* using either of these methods:
   - Using the CCA:
     1) Select the DB2 database alias that you want to add as a data source.
     2) Click on the **Properties** push button.
     3) Select the **Register this database for ODBC** check box.
   - Using the ODBC Driver Manager:
     1) Run the ODBC Driver Manager, as described in its documentation. This is usually done in one of two ways:
        – Double-click on the **ODBC** Folder in OS/2, and double-click on the **ODBC Administrator** icon.
        – Run **odbcadm.exe** from the command line.
     2) Click on the **Add** push button from the Data Sources window. The Add Data Source Window opens.
     3) Double-click on the IBM DB2 ODBC DRIVER in the list.
     4) Select the DB2 database to add and click on **OK**.

3. Configure the DB2 CLI/ODBC driver using the CCA: (Optional)
   a. Select the DB2 database alias you want to configure.
   b. Click on the **Properties** push button. The Database Properties window opens.
   c. Click on the **Settings** push button. The CLI/ODBC Settings window opens.
   d. Click on the **Advanced** push button. You can set the configuration keywords in the window that appears. These keywords are associated with the database *alias name*, and affect all DB2 CLI/ODBC applications that access the database. The online help explains all of the keywords, as does and the *Installation and Configuration Supplement* manual.
4. If you have installed ODBC access (as described above), you can now access DB2 data using ODBC applications. Start the ODBC application and go to the Open window. Select the **ODBC databases** file type. The DB2 databases that you added as ODBC data sources will be selectable from the list. Many ODBC applications will open the table as read-only unless a unique index exists.

Detailed Configuration Information

The section "Platform Specific Details for CLI/ODBC Access" on page 61 should provide you with all of the information you require. *Installation and*

Configuration Supplement includes additional information on setting up and using DB2 CLI and ODBC applications. (The online *Installation and Configuration Supplement* manual is located in the directory *x*:\doc\en\html, where *x:* is the letter that designates your CD-ROM and en is the two-character country code that represents your language, for example en for English.) This information is useful where DB2 tool support is not available, and for administrators who require more detailed information.

The following topics are covered in the *Installation and Configuration Supplement* online manual:

- How to Bind the DB2 CLI/ODBC Driver to the Database
- How to Set CLI/ODBC Configuration Keywords
- Configuring db2cli.ini

Running Java Programs

You can develop Java programs to access DB2 databases with the appropriate Java Development Kit (JDK) on AIX, HP-UX, Linux, OS/2, PTX, Silicon Graphics IRIX, Solaris Operating Environment, or Windows 32-bit operating systems. The JDK includes Java Database Connectivity (JDBC), a dynamic SQL API for Java.

For DB2 JDBC support, you must include the DB2 Java Enablement component when you install the DB2 client. With DB2 JDBC support you can build and run JDBC applications and applets. These contain dynamic SQL only, and use a Java call interface to pass SQL statements to DB2.

The DB2 Application Development Client provides support for Java embedded SQL (SQLJ). With DB2 SQLJ support and DB2 JDBC support you can build and run SQLJ applications and applets. These contain static SQL and use embedded SQL statements that are bound to the DB2 database.

Java can also be used on the server to create JDBC and SQLJ stored procedures and user-defined functions (UDFs).

Building and running different types of Java programs requires support from different components of DB2:

- To build JDBC applications, you must install a DB2 client with the DB2 Java Enablement component. To run JDBC applications, your DB2 client with the DB2 Java Enablement component must connect to a DB2 server.
- To build SQLJ applications, you must install the DB2 Application Development Client and a DB2 Administrative Client with the DB2 Java Enablement component. To run SQLJ applications, your DB2 client with the DB2 Java Enablement component must connect to a DB2 server.

- To build JDBC applets, you must install a DB2 client with the DB2 Java Enablement component. To run JDBC applets, the client machine does not require any DB2 components.
- To build SQLJ applets, you must install the DB2 Application Development Client and a DB2 Administrative Client with the DB2 Java Enablement component. To run SQLJ applets, the client machine does not require any DB2 components.

For detailed information on building and running JDBC and SQLJ programs refer to the *Application Building Guide*. For more information on DB2 programming in Java, refer to the *Application Development Guide*. This covers creating and running JDBC and SQLJ applications, applets, stored procedures and UDFs.

For the latest, updated DB2 Java information, visit the Web site at: http://www.ibm.com/software/data/db2/java

Configuring the Environment

To build DB2 Java programs, you need to install and configure the appropriate version of a Java Development Kit (JDK) on your development machine. To run DB2 Java applications, you must install and configure the appropriate version of either a Java Runtime Environment (JRE) or JDK on your development machine. The following table lists the version of the JDK that is appropriate for your development machine:

AIX The IBM AIX Developer Kit, Java Technology Edition, Version 1.1.8. On AIX systems that do not have a JDK installed, this JDK is automatically installed with the DB2 Application Development Client.

HP-UX
The HP-UX Developer's Kit for Java, Release 1.1.8, from Hewlett-Packard.

Linux The IBM Developer Kit for Linux, Java Technology Edition, Version 1.1.8.

OS/2 The IBM Java Development Kit for OS/2, version 1.1.8, which is available on the product CD-ROM.

PTX The ptx/JSE, Version 1.2.1, from IBM.

SGI IRIX
The Java 2 Software Development Kit for SGI IRIX, version 1.2.1, from SGI.

Solaris Operating Environment
The Java Development Kit for Solaris, version 1.1.8, from Sun Microsystems.

Windows 32-bit operating systems
The IBM Developer Kit for Windows 32-bit operating systems, Java Technology Edition, Version 1.1.8. When you install the DB2 Application Development Client, this JDK is automatically installed in the sqllib\java\jdk directory.

For information on installing and configuring any of the above JDKs, please refer to: http://www.ibm.com/software/data/db2/java

For all supported platforms, you must also install and configure a DB2 client with the DB2 Java Enablement component. To bind SQLJ programs to a database, you must install and configure a DB2 Administrative Client with the DB2 Java Enablement component.

To run DB2 Java stored procedures or UDFs, you also need to update the DB2 database manager configuration to include the path where the JDK version 1.1 is installed on your development machine. You can do this by entering the following on the command line:

You can check the DB2 database manager configuration to verify the correct value for the JDK11_PATH field by entering the following command:

```
db2 get dbm cfg
```

You may want to pipe the output to a file for easier viewing. The JDK11_PATH field appears near the beginning of the output. For more information on these commands, refer to the *Command Reference*.

On Solaris Operating Environment, some Java Virtual Machine implementations do not work well in programs that run in a "setuid" environment. The shared library that contains the Java interpreter, `libjava.so`, may fail to load. As a workaround, you can create symbolic links for all needed JVM shared libraries in /usr/lib, with a command similar to the following (depending on where Java is installed on your machine):

```
ln -s /opt/jdk1.1.3/lib/sparc/native_threads/*.so /usr/lib
```

For more information on this and other workarounds available, please visit: http://www.ibm.com/software/data/db2/java

To run Java programs, the following environment variables are automatically updated during DB2 installation on OS/2 and the Windows Operating System, and during instance creation on UNIX platforms.

On UNIX platforms:

- CLASSPATH includes "." and the file sqllib/java/db2java.zip

- On AIX, Linux, PTX, Silicon Graphics IRIX, and Solaris Operating Environment: LD_LIBRARY_PATH includes the directory sqllib/lib
- On HP-UX: SHLIB_PATH includes the directory sqllib/lib
- On Solaris Operating Environment only: THREADS_FLAG is set to ″native″

On Windows and OS/2 platforms:

- CLASSPATH includes ″.″ and the file %DB2PATH%\java\db2java.zip

In order to build and run SQLJ programs, CLASSPATH is also automatically updated to include these files:

On UNIX platforms:

- sqllib/java/sqlj.zip (required to build SQLJ programs)
- sqllib/java/runtime.zip (required to run SQLJ programs)

On Windows and OS/2 platforms:

- %DB2PATH%\java\sqlj.zip (required to build SQLJ programs)
- %DB2PATH%\java\runtime.zip (required to run SQLJ programs)

Java Applications

Start your application from the desktop or command line by running the Java interpreter on the executable program with this command:

```
java prog_name
```

where prog_name is the name of the program.

The DB2 JDBC driver handles the JDBC API calls from your application and uses the DB2 client to communicate the requests to the server and receive the results. An SQLJ application must be bound to the database before it is run.

Java Applets

Because Java applets are delivered over the web, a web server must be installed on your DB2 machine (server or client).

To run your applet, make sure your .html file is properly configured. Start the JDBC applet server on the TCP/IP port specified in the `.html` file. For example, if you specified:

```
param name=port value='6789'
```

then you would enter:

```
db2jstrt 6789
```

You must ensure that your working directory is accessible to your web browser. If it is not, copy your applet's .class and .html files into a directory that is accessible. For SQLJ applets, you must also copy the profile .class and .ser files as well.

Copy the sqllib/java/db2java.zip file into the same directory as these other files. For SQLJ applets, also copy the sqllib/java/runtime.zip file into this directory. Then on your client machine start your web browser (which supports JDK 1.1) and load the .html file.

When your applet calls the JDBC API to connect to DB2, the JDBC driver establishes separate communications with the DB2 database through the JDBC applet server residing on the DB2 server. An SQLJ applet must be bound to the database before it is run.

Part 2. Reference and Troubleshooting

Chapter 6. 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 host or AS/400 database server that DB2 Connect accesses.
- The *database connection services* (DCS) directory, which contains information specific to host or AS/400 database server databases.
- The *system database directory*, which contains name, node, and authentication information for every database that DB2 Connect accesses.

Notes:

1. Before updating these directories, you should configure communications on the host or AS/400 database server and workstations. For more information, refer to the *Installation and Configuration Supplement*.
2. On OS/2 and Windows 32-bit operating systems, database directories can be updated using the DB2 Universal Database Client Configuration Assistant (CCA).

   On all other platforms the database directories must be updated using the DB2 Command Line Processor (CLP).
3. "Updating the Directories" on page 84 provides example command syntax. For further information, refer to the *Command Reference*.
4. If you are using DCE for each host or AS/400 database 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 D. Using DCE Directory Services" on page 199 and the *Administration Guide*. This section assumes that you are *not* using DCE Directory Services.

Collecting Information

"Appendix B. Directory Customization Worksheet" on page 193 shows the information that you need to collect. You may find it convenient to make a copy of the worksheet and enter your system values.

Node Directory

You can specify the following information in the node directory:

Node name
A nickname for the host or AS/400 database 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 TCPIP.

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 if there is a mismatch between upper and lower case names).

Security type
The type of security checking that will be done. For APPC nodes, the valid options are `SAME`, `PROGRAM`, and `NONE`. 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 10. Security" on page 111, and refer to the *Command Reference*.

TCP/IP remote hostname or IP address
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.

For DB2 for OS/390 remote hosts, the hostname appears in the DSNL004I message (DOMAIN=hostname) when the Distributed Data Facility (DDF) is started.

TCP/IP service name or port number
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.

For DB2 for OS/390 remote hosts, the port number is defined in the Boot Strap Data Set (BSDS) as PORT and is also provided in the DSNL004I message (TCPPORT=portnumber) when the Distributed Data Facility (DDF) is started.

Note: A second port used for two-phase commit resynchronization operations over TCP/IP connections is assigned by the server. For example, the DB2 Universal Database for OS/390 bootstrap dataset assigns a port number (RESPORT) to be used for resynchronization for inbound connections to DB2 Universal Database 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 host or AS/400 database server. 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 host or AS/400 database server system, as follows:

MVS/ESA

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

The LOCATION NAME can be determined by logging in to TSO and issuing the following SQL query using one of the available query tools:

```
select current server from sysibm.sysdummy1
```

LOCATION NAME is also defined in the MVS/ESA Boot Strap Data Set (BSDS) as well as the DSNL004I message (LOCATION=location), which is written when the Distributed Data Facility (DDF) is started.

OS/390

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

The LOCATION NAME can be determined by logging in to TSO and issuing the following SQL query using one of the available query tools:

```
select current server from sysibm.sysdummy1
```

LOCATION NAME is also defined in the Boot Strap Data Set (BSDS) as well as the DSNL004I message (LOCATION=location), which is written when the Distributed Data Facility (DDF) is started.

VSE or VM
The database name (DBNAME)

OS/400
The relational database name (RDBNAME)

Other For OS/2, Windows NT, Windows 2000, and UNIX-based systems, the database alias found in the database directory.

Application requester name
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.

Format: AR <application_requester_name>

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. The parameter string cannot be set using the Client Configuration Assistant, and that when using the CLP the parameter string must be enclosed in either single quotes (for example, on OS/2 or Windows NT), or in double quotes (for example, on AIX):

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 11. SQLCODE Mapping" on page 121.

,D This is the second positional parameter. If it is specified the application will disconnect from the host or AS/400 database server database when 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, refer to the *Message Reference*.

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

„INTERRUPT_ENABLED

This is the third positional parameter. 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 host or AS/400 database 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, Windows NT, and Windows 2000.

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 host or AS/400 database server, in order to process additional database requests. On platforms other than AIX V4.1 and later, SNA Server V3.1 and later, OS/2, Windows NT and Windows 2000, 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 Universal Database 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.

,,,,,SYSPLEX

This parameter, the 6th positional parameter, can be used to explicitly enable DB2 Connect SYSPLEX support for a particular database.

A new profile (environment or registry) variable has also been introduced, called `DB2SYSPLEX_SERVER`, and it can be used to disable the SYSPLEX support at the workstation level.

,,,,,LOCALDATE="<value>"

This parameter, the seventh positional parameter, is used to enable DB2 Connect date formatting support. This is implemented using a date mask for the *<value>* as follows:

Suppose you issue the following CLP (command line processor) statements:

```
catalog appc node nynode remote nycpic security program
catalog dcs database nydb1 as new_york
catalog database nydb1 as newyork1 at node nynode
     authentication dcs
```

The database alias *newyork1* is to be used for accessing a host database without date transformation because no date mask has been specified.

However, with the new date formatting support, you can now use the following CLP commands. In this case, because the CLP is being used, and the parameter string is itself being specified using double quotes, the LOCALDATE value has to be specified inside two pairs of double quotes. Note the use of the operating system escape character "\" (backslash) to ensure that the double quotes are not stripped from the LOCALDATE specification. See also "Specifying the Parameter String" on page 82.

```
catalog dcs database nydb2 as new_york
     parms \",,,,,,LOCALDATE=\"\"YYYYMMDD\"\"\"
catalog database nydb2 as newyork2 at node nynode
     authentication dcs
```

The database alias "newyork2" gives you access to the same host database but, in addition, it has a date format mask specified. This example illustrates that

the date format mask is specified using the keyword LOCALDATE and is the seventh positional parameter in the PARMS field of a DCS directory entry.

For the date mask to be valid, ALL of the following must be true:

1. There can only be at most one sequence each of Y's, M's, and D's where Y is a year digit, M is a month digit, and D is a day digit.
2. The maximum number of Y's in a sequence is 4.
3. The maximum number of M's in a sequence is 2.
4. The maximum number of D's in a sequence is 2.

For instance, the following are all valid date masks:

```
"YYyyMmDd"   - Y, M, and D digits are case-insensitive
"MM+DD+YYYY" - OK to have a mask longer than 10 bytes
               and to have characters other than Y, M,
               and D in the mask
"abcYY+MM"   - OK not to have a sequence of D's
```

The following are all invalid date masks:

```
"YYYYyMMDD"  - invalid there are 5 Y's in a sequence
"YYYYMDDM"   - invalid there are 2 sequences of M's
```

If a date format mask is invalid, no error will be issued. It will just be ignored. Just because a date mask is valid does not mean it will be used. Date format transformation based on a valid date mask will only be performed if ALL of the following are true:

1. There is no SQL error.
2. The output is a date value in ISO-like (ISO and JIS) format.
3. The output data area is at least 10 bytes long. This is the minimum size of an output data area in order for a data value to be stored there even if NO date format transformation is to be performed. This requirement applies even if the date format mask ends up being shorter than 10 bytes.
4. There is a valid date format mask specified in the DCS directory entry and this mask fits in the output data area.

,,,,,,CHGPWD_SDN=<name>

This parameter, the eighth positional parameter, is used to specify the symbolic destination name to be

used for Password Expiration Management (PEM). The value specified for <name> is case sensitive.

"Changing Your MVS Password" on page 116 shows an example of cataloging a dcs database directory using CHGPWD_SDN, as follows:

```
catalog dcs database db1 as dsn_db_1 parms
   ",,,,,,,CHGPWD_SDN=pempgm"
```

,,,,,,,BIDI=<ccsid>

This parameter, the ninth positional parameter, is used to specify the Bidirectional (BiDi) CCSID to be used to override the default server database BiDi CCSID. For example:

```
",,,,,,,,BIDI=xyz"
```

where *xyz* represents the CCSID override (see 1 on page 81).

For a list of what BiDi CCSIDs are supported along with their string types, refer to the *Administration Guide*.

The following BiDi attributes are required for correct handling of BiDi data on different platforms:

- Numeral shape (ARABIC vs HINDI)
- Orientation (RIGHT-TO-LEFT vs LEFT-TO-RIGHT)
- Shaping (SHAPED vs UNSHAPED)
- Symmetric swapping (YES or NO)
- Text type (LOGICAL vs VISUAL)

Since defaults on different platforms are not the same, problems appear when DB2 data is sent from one platform to another. For example, Windows platforms use LOGICAL UNSHAPED data, while data on MVS and OS/390 is usually in SHAPED VISUAL format. Therefore, without any support for BiDi attributes, data sent from DB2 for MVS or OS/390 to DB2 Connect on Windows displays incorrectly.

When data is exchanged between DB2 Connect and a database on a server, it is usually the receiver that performs conversion on the incoming data. The same convention would normally apply to BiDi layout transformation also, which is in addition to the usual code page conversion. However, currently no host DB2 product supports BiDi-specific CCSIDs or BiDi layout transformation. Therefore, DB2 Connect has been enhanced with the optional ability to perform BiDi layout transformation on data it is about to send to the server database in addition to data received from the server database.

For DB2 Connect to perform BiDi layout transformation on outgoing data to a server database, the BiDi CCSID of the server database will have to be overridden (see 2). This is accomplished through the use of the BIDI parameter in the PARMS field of the DCS database directory entry for the server database.

The use of this feature is best illustrated with an example.

Consider a Hebrew DB2 client running CCSID 62213 (BiDi string type 5) and you would like to access a DB2 host database running CCSID 424 (BiDi string type 4). However, you know that the data contained in the DB2 host database is instead based on CCSID 8616 (BiDi string type 6).

There are two problems in this situation. The first is that the DB2 host database does not know the difference between the BiDi string types with CCSIDs 424 and 8616. The second problem is that the DB2 host database does not recognize the DB2 client CCSID of 62213. It only supports CCSID 862, which is based on the same code page as CCSID 62213.

You will need to make sure that data sent to the DB2 host database is in BiDi string type 6 format to begin with and also let DB2 Connect know that it has to perform BiDi layout transformation on data it receives from the DB2 host database. You will use the following cataloging for the DB2 host database:

```
catalog dcs database nydb1 as TELAVIV parms ",,,,,,,,BIDI=8616"
```

This tells DB2 Connect to override the DB2 host database CCSID of 424 with 8616. This override includes the following processing:

1. DB2 Connect will connect to the DB2 host database using CCSID 862.
2. DB2 Connect will perform BiDi layout transformation on data it is about to send to the DB2 host database from CCSID 62213 (BiDi string type 5) to CCSID 62221 (BiDi string type 6).
3. DB2 Connect will perform BiDi layout transformation on data it receives from the DB2 host database from CCSID 8616 (BiDi string type 6) to CCSID 62213 (BiDi string type 5).

Notes:

1. The environment variable or registry value DB2BIDI has to be set to YES in order for the BIDI parameter to take effect.
2. If you would like DB2 Connect to perform layout transformation on data it is about to send to the DB2 host database even though you do not have to override its CCSID, you still have to add the BIDI parameter in the DCS database directory PARMS field. In this case, the CCSID that you should provide would be the default DB2 host database CCSID.
3. In some cases, use of a bidirectional CCSID may cause the SQL query itself to be modified such that it is not recognized by the DB2 server.

Specifically, you should try to avoid using IMPLICIT CONTEXTUAL and IMPLICIT RIGHT-TO-LEFT CCSIDs when a different string type can be used. CONTEXTUAL CCSIDs can produce unpredictable results if the SQL query contains quoted strings. Avoid using quoted strings in SQL statements, and use host variables instead when possible.

If a specific bidirectional CCSID is causing problems which cannot be rectified by following these recommendations, then you should set the environment variable or registry value DB2BIDI to NO.

Specifying the Parameter String

Here are examples of some parameter strings you could specify.

For example, you could specify any of the following where "\" (backslash) is the operating system escape character:

On AIX:

```
NOMAP
/u/username/sqllib/map/dcs1new.map,D
,D
,,INTERRUPT_ENABLED
NOMAP,D,INTERRUPT_ENABLED,,,SYSPLEX,LOCALDATE=\"\"YYMMDD\"\",,
```

On OS/2, Windows NT, or Windows 2000:

```
NOMAP
d:\sqllib\map\dcs1new.map,D
,,INTERRUPT_ENABLED
NOMAP,D,INTERRUPT_ENABLED,,,SYSPLEX,LOCALDATE=\"\"YYMMDD\"\",,
```

Alternatively you can accept the defaults by not specifying a parameter string.

Note: Because of the need to specify two pairs of double quotes when specifying the LOCALDATE mask in the parameter string, you must use the operating system escape character "\" (backslash), for example:

```
db2 catalog dcs db x as y parms \",,,,,,LOCALDATE=\"\"YYMMDD\"\"\"
```

This results in the following DCS directory entry:

```
DCS 1 entry:

 Local database name               = X
 Target database name              = Y
 Application requestor name        =
 DCS parameters                    = ,,,,,,LOCALDATE="YYMMDD"
 Comment                           =
 DCS directory release level       = 0x0100
```

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 host or AS/400 database server. 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`, `SERVER_ENCRYPT`, `CLIENT`, `DCE`, `DCS`, and `DCS_ENCRYPT`. For more information, see "Chapter 10. Security" on page 111.

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 host or AS/400 database 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 CATALOG command on any DB2 Connect system or the Add Database Wizard of the CCA on OS/2 and Windows 32-bit operating systems. If you have the DB2 Application Development Client, you can also create an application program to catalog entries. For information about APIs, refer to the *Administrative API Reference* and the *Command Reference*.

Note: To catalog a database, you must have *sysadm* or *sysctrl* 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 Universal Database for OS/390 Version 5.1 or a DB2 Universal Database for AS/400 Version 4.2 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
     ```

     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, Windows NT, or Windows 2000:

   ```
   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"'
   ```

Note: See "Specifying the Parameter String" on page 82 for information about using the operating system escape character when specifying the LOCALDATE mask in the parameter string.

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, refer to the appropriate *DB2 Connect Quick Beginnings* book.

Chapter 7. 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 host or AS/400 database server. For more information about binding, refer to the *Application Development Guide*.

Because several of the utilities supplied with DB2 Connect are developed using embedded SQL, they must be bound to a host or AS/400 database server before they can be used with that system. If you do not use the DB2 Connect utilities and interfaces listed in Table 4 on page 90, you do not have to bind them to each of your host or AS/400 database 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 is installed, the DB2 Connect utilities must be bound to each host or AS/400 database 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 Universal Database for OS/390 via a DB2 Connect Enterprise Edition for Window NT server, 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 server.

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 E. Binding Utilities for Back-Level Clients" on page 209 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 host or AS/400 database server that you are binding to, do the following:

1. Make sure that you have sufficient authority for your host or AS/400 database server 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 host or AS/400 database server, ddcsmvs.lst is the bind list file for MVS, and *path* represents 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* represents the logical drive where DB2 Connect was installed and *INSTHOME* represents 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 4 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 4. 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						

Table 4. Bind Files and Packages (continued)

Component	Bind file	Package	MVS or OS/390	VSE	VM	OS/400
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

To determine these values for DB2 Connect 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.

For more information on bind options, refer to the *Command Reference*.

Notes:

a. Using the bind option `sqlerror continue` is required; however, this option is automatically specified for you when you bind applications using the DB2 tools or the command line processor. 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 host or AS/400 database 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 host or

AS/400 database server. Also, to connect to a DB2 Universal Database database, we recommend that you use the DB2 clients 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 E. Binding Utilities for Back-Level Clients" on page 209 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 host or AS/400 database server through DB2 Connect. Ensure that you are referring to the DRDA-specific description.

Note: Some parameters of the BIND command may not be supported by your host or AS/400 database server. For more information, refer to the documentation supplied with your host or AS/400 database server RDBMS.

Rebinding

After you bind your application (and create the package on the host or AS/400 database 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 Universal Database 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 host or AS/400 database 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 8. Database System Monitor

This chapter summarizes the capabilities of the DB2 System Monitor for DB2 Connect users. The following System Monitor enhancements are provided in DB2 Connect Version 7:

- Snapshot monitoring. A snapshot of the system gives you information for a specific point in time. A snapshot is a picture of the current state of activity in the database manager for a particular object or group of objects. There are five basic snapshots of DCS database information available through the monitor.
- A graphical user interface equivalent of the LIST DCS APPLICATIONS CLP command. The LIST command provides a more concise reading of the state of the system than the snapshot. Graphical LIST capability is provided through the DB2 Command Center. See "LIST DCS APPLICATIONS EXTENDED" on page 102

For more information about the database system monitor, refer to the *System Monitor Guide and Reference*.

Monitoring Connections for Remote Clients

You can use the Database System Monitor with DB2 Connect Enterprise Edition to monitor the remote client connections. To monitor clients that are local to the DB2 Connect server, that are running on the server itself, you will need to set the following environment variable:

```
db2set DB2CONNECT_IN_APP_PROCESS=NO
```

For example, when an error occurs at the host or AS/400 system, 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 information 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.

Turning on Monitor Switches for DB2 Connect

The system monitor is always active. However, if you want to get more detail in GET SNAPSHOT output, you should turn on the corresponding monitor switches. The monitor switches relevant to DB2 Connect are STATEMENT (for statement-level information), and UOW (for transaction-level information).

To change monitor switches, use the **db2 update monitor switches** command. Refer to the *Command Reference* for the syntax of this command. Here is an example, which creates DB2 System Monitor statistics for Units of Work (UOW):

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

Listing the Status of Monitor Switches

To list the status of monitor switches, use the **db2 get monitor switches** command.

Using the GET SNAPSHOT Commands

The DB2 monitor maintains a running tally of valuable system information. You can get a summary of system status at any time by issuing the GET SNAPSHOT command. You can take monitor snapshots if you have SYSMAINT, SYSCTRL, or SYSADM authority for the database manager instance that you wish to monitor.

There are five snapshot commands useful for monitoring DCS information. They are:

- GET SNAPSHOT FOR ALL DCS DATABASES
- GET SNAPSHOT FOR ALL DCS APPLICATIONS
- GET SNAPSHOT FOR DCS APPLICATION ...
- GET SNAPSHOT FOR DCS DATABASE ON db_alias
- GET SNAPSHOT FOR DCS APPLICATIONS ON db_alias

Each snapshot command will produce a detailed report about the area you requested.

For instance, issuing the GET SNAPSHOT FOR DCS DATABASE ON DCSDB will produce the following report:

```
             DCS Database Snapshot

DCS database name                       = DCSDB
Host database name                      = GILROY
First database connect timestamp        = 12-15-1999 10:28:24.596495
Most recent elapsed time to connect     = 0.950561
Most recent elapsed connection duration = 0.000000
Host response time (sec.ms)             = 0.000000
Last reset timestamp                    =
```

```
Number of SQL statements attempted       = 2
Commit statements attempted              = 1
Rollback statements attempted            = 0
Failed statement operations              = 0
Total number of gateway connections      = 1
Current number of gateway connections    = 1
Gateway conn. waiting for host reply     = 0
Gateway conn. waiting for client request = 1
Gateway communication errors to host     = 0
Timestamp of last communication error    = None
High water mark for gateway connections  = 1
Rows selected                            = 0
Outbound bytes sent                      = 140
Outbound bytes received                  = 103
```

This report provides information on database connections, performance, errors and throughput of SQL requests. DB2 Monitor snapshots can be much more detailed, in fact. For instance, if you issue the GET SNAPSHOT FOR ALL DCS APPLICATIONS command, you will receive a report similar to the following:

```
           DCS Application Snapshot

Client application ID                    = 09150F74.B6A4.991215152824
  Sequence number                        = 0001
  Authorization ID                       = SMITH
  Application name                       = db2bp
  Application handle                     = 1
  Application status                     = waiting for request
  Status change time                     = 12-15-1999 10:29:06.707086
  Client node                            = sys143
  Client release level                   = SQL06010
  Client platform                        = AIX
  Client protocol                        = TCP/IP
  Client codepage                        = 850
  Process ID of client application       = 49074
  Client login ID                        = smith
  Host application ID                    = G9150F74.B6A5.991215152825
  Sequence number                        = 0000
  Database alias at the gateway          = MVSDB
  DCS database name                      = DCSDB
  Host database name                     = GILROY
  Host release level                     = DSN05012
  Host CCSID                             = 500

Outbound communication address           = 9.21.21.92 5021
Outbound communication protocol          = TCP/IP
Inbound communication address            = 9.21.15.116 46756
First database connect timestamp         = 12-15-1999 10:28:24.596495
Host response time (sec.ms)              = 0.000000
Time spent on gateway processing         = 0.000000
Last reset timestamp                     =
Rows selected                            = 0
Number of SQL statements attempted       = 2
```

```
Failed statement operations               = 0
Commit statements                         = 1
Rollback statements                       = 0
Inbound bytes received                    = 404
Outbound bytes sent                       = 140
Outbound bytes received                   = 103
Inbound bytes sent                        = 287
Number of open cursors                    = 0
Application idle time                     = 1 minute and 32 seconds

UOW completion status                     =
Previous UOW completion timestamp         = 12-15-1999 10:28:25.592631
UOW start timestamp                       = 12-15-1999 10:29:06.142790
UOW stop timestamp                        =
Elapsed time of last completed uow (sec.ms)= 0.034396

Most recent operation                     = Execute Immediate
Most recent operation start timestamp     = 12-15-1999 10:29:06.142790
Most recent operation stop timestamp      = 12-15-1999 10:29:06.707053

Statement                                 = Execute Immediate
Section number                            = 203
Application creator                       = NULLID
Package name                              = SQLC2C07
SQL compiler cost estimate in timerons    = 0
SQL compiler cardinality estimate         = 0
Statement start timestamp                 = 12-15-1999 10:29:06.142790
Statement stop timestamp                  = 12-15-1999 10:29:06.707053
Host response time (sec.ms)               = 1.101612
Elapsed time of last completed stmt(sec.ms)= 0.564263
Rows fetched                              = 0
Time spent on gateway processing          = 0.013367
Inbound bytes received for statement      = 220
Outbound bytes sent for statement         = 130
Outbound bytes received for statement     = 49
Inbound bytes sent for statement          = 27
SQL statement text:
create table t12 (col1 int, col2 char)
```

For more information about the GET SNAPSHOT command and other useful DB2 Monitor commands, refer to the *System Monitor Guide and Reference*.

Listing DCS Application Status

As of DB2 Connect V5.2, the System Monitor provides three forms of the `LIST DCS APPLICATIONS` command, as follows:

- LIST DCS APPLICATIONS
- LIST DCS APPLICATIONS SHOW DETAIL
- LIST DCS APPLICATIONS EXTENDED.

LIST DCS APPLICATIONS

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 7 to DB2 Universal Database for OS/390):

```
Auth Id  Application Name     Appl.      Host 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 7 to DB2 Universal Database for OS/390):

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

Auth.Id
The authorization ID that was used to log on to the host or AS/400 database 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 165.) The agent ID is also required when using the FORCE USERS command or API.

Host 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 host or AS/400 database server. You can use this element in conjunction with the Application ID to correlate the client and server parts of the application information.

LIST DCS APPLICATIONS SHOW DETAIL

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

```
Auth Id  Application Name     Appl.      Client Application Id            Seq# Client
                              Handle                                           DB Alias
-------- -------------------- ---------- -------------------------------- ---- --------
NEWTON   db2bp                0          09151251.07D3.980925183850       0001 MVSDB

 Client   Client   Client     Host Application Id              Seq# Host DB Name
 Node     Release  Codepage
 -------- -------- ---------- -------------------------------- ---- --------------------
 antman   SQL05020 819        G9151251.G7D4.980925183851       0000 GILROY

Host
Release
--------
DSN05011
```

This report is unformatted, and for this reason you may find the "LIST DCS APPLICATIONS EXTENDED" on page 102 report more useful.

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, refer to the *Administration Guide*.

This value lets you correlate connections from clients to the DB2 Connect workstation and from the DB2 Connect workstation to the host or AS/400 database 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.
- SQL05020 for Version 5.2 of DB2 Universal Database and DB2 Connect products and their clients.
- SQL06010 for Version 6.1 of DB2 Universal Database and DB2 Connect products and their clients.
- SQL07010 for Version 7 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 host or AS/400 database server databases, the host or AS/400 database 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 host or AS/400 database server product (for example, DSN for DB2 Universal Database for OS/390, ARI for DB2 for VSE & VM, or QSQ for DB2 Universal Database for AS/400)

VV Represents a two-digit version number, such as 01.

RR Represents a two-digit release number.

M Represents a one-digit modification level.

LIST DCS APPLICATIONS EXTENDED

You can use the LIST DCS APPLICATIONS command with the option EXTENDED in order to generate an Extended Report. The Extended Report lists all the fields that are listed when the SHOW DETAIL option is specified on the command, plus nine new fields:

- DCS application status
- Status change time
- Client platform
- Client protocol
- Host Coded Character Set Identifier (CCSID).
- Client login ID
- Process ID of client application
- Database alias at the gateway
- DCS database name

While the existing command options list the fields horizontally, one line per application, the new option lists them vertically, one field per line.

Here is the new syntax of the command:

```
LIST DCS APPLICATIONS [SHOW DETAIL | EXTENDED ]
```

And here is sample output from this command, when using the new option EXTENDED:

```
                List of DCS Applications - Extended Report

Client application ID                     = 09151251.0AA7.981015204853
  Sequence number                         = 0001
  Authorization ID                        = NEWTON
  Application name                        = db2bp
  Application handle                      = 1
  Application status                      = waiting for request
```

```
Status change time                      = 10-15-1998 16:50:29.489160
Client node                             = antman
Client release level                    = SQL05020
Client platform                         = AIX
Client protocol                         = TCP/IP
Client codepage                         = 819
Process ID of client application        = 39324
Client login ID                         = smith
Host application ID                     = G9151251.GAA8.981015204854
Sequence number                         = 0000
Database alias at the gateway           = MVSDB
DCS database name                       = DCSDB
Host database name                      = GILROY
Host release level                      = DSN05011
Host CCSID                              = 500
```

The application status field contains one of the following three values:

1. `connect pending - outbound.` This means that the request to connect to a host database has been issued, and DB2 Connect is waiting for the connection to be established.
2. `waiting for request.` This means that the connection with the host database has been established, and that DB2 Connect is waiting for an SQL statement from the client application
3. `waiting for reply.` This means that the SQL statement has been sent to the host database.

Also, the status change time is only shown in the report if the System Monitor UOW switch was turned on during processing. Otherwise, ″Not Collected″ will be shown.

Using the DB2 Control Center to List Extended DCS Applications Information

You can use the DB2 Version 7 Control Center to perform DB2 Connect gateway monitoring. This section shows how you can use the Control Center to display the same report as provided by the **list dcs applications extended** command.

To view the extended report for any application:

1. Expand the tree under the **systems** icon of the Control Center to display **System —> Instances —> Gateway Connections**. If you right-click the mouse on any instance under the **Gateway Connections** folder, a pop-up menu will appear. Select the Applications... item from this menu. The Applications window appears. This window has a tabbed notebook appearance, with one tab titled **Applications**. If there are gateway applications in your instance, there will be a second tab titled **Gateway Applications**.

2. The main window of each page contains columns of information corresponding to the fields of the LIST DCS APPLICATIONS EXTENDED report. The first six columns, visible in the window, provide the following data:

   ```
   Client Node
   Application Name
   Client Application ID
   Host Application ID
   Database Alias at Gateway
   Status
   ```

   The rest of the fields of the report can be viewed by moving the horizontal scroll bar at the bottom of the window.

All of the fields listed by the LIST DCS APPLICATIONS EXTENDED command are present in this view.

Using the Windows Performance Monitor

Windows NT and Windows 2000 provides a useful tool for monitoring the performance of your DB2 applications. The Performance Monitor, which is one of the Windows administrative tools, displays a graphical representation of system performance. You can choose a variety of system, database, and communications-related items to monitor and map them together in a graphical representation.

For example, the reports available through the **GET SNAPSHOT FOR ALL DCS DATABASES** or **GET SNAPSHOT FOR ALL DCS APPLICATIONS** commands can be graphed in real time using the monitor, and compared directly with values such as CPU usage. You can directly compare the effects of different settings on database or communications performance. You can save your specialized configurations of settings in PMC files that you can later retrieve.

For example in the figure below, several DB2 measures are being graphed against CPU usage. The collection of values being charted was saved in the file db2chart.pmc. You may save as many PMC files as you like, each reflecting a different cross-section of system performance.

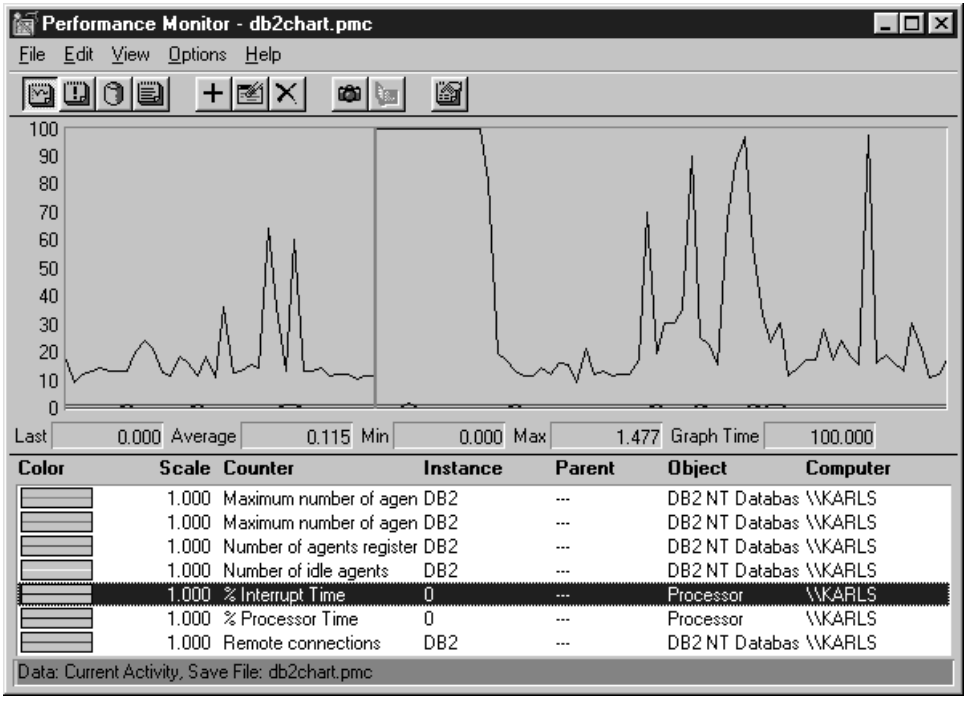


Figure 5. Performance Monitor

To enable monitoring of local applications you will need to turn the DB2CONNECT_IN_APP_PROCESS environment variable off.

For more DCS APPLICATIONS commands, or for more information regarding using the Windows Performance Monitor with DB2 UDB and DB2 Connect, refer to the *Administration Guide*.

Chapter 9. Administration Utilities

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

Command Line Processor

The command line processor lets you issue SQL statements against a host or AS/400 database server, preceded by db2. For differences between host and AS/400 SQL and DB2 Connect SQL, see "Host or AS/400 Server SQL Statements Supported by DB2 Connect" on page 53 and "Host or AS/400 Server SQL Statements Rejected by DB2 Connect" on page 54.

To invoke the command line processor in interactive input mode, do the following:

OS/2 Click **OS/2 Warp**, and select **IBM DB2 —> Command Line Processor** or enter the **db2** command.

Windows 32-bit operating systems
Click **Start**, and select **Programs —> IBM DB2 —> Command Line Processor**.

Note: You can also invoke the command line processor in interactive input mode by entering the **db2cmd** command followed by the **db2** command at an MS-DOS prompt.

UNIX Enter the **db2** command from the command line processor

In interactive input mode, the prompt looks like this:

```
db2 =>
```

In interactive input mode, you do not have to enter DB2 commands with a db2 prefix; instead, you just enter the DB2 command. For example:

```
db2 => list database directory
```

To enter operating system commands in interactive mode, precede the operating-system command with an exclamation mark (!). For example:

```
db2 => !dir
```

If you need to enter a long command that does not fit on one line, enter a space followed by the line continuation character, \. When you have reached the end of the line, press the **Enter** key to continue entering the command on the next line. For example:

```
db2 => select empno, function, firstname, lastname, birthdate, from \
db2 (cont.) => employee where function='service' and \
db2 (cont.) => firstname='Lily' order by empno desc
```

To end interactive input mode, enter the **quit** command.

Before you can use the command line processor, it must be bound, as described in "Chapter 7. Binding Applications and Utilities" on page 87.

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, refer to the *Command Reference*.

Using Import and Export Utilities

The import and export utilities let you move data from a S/390 host or AS/400 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 OS/390 into a delimited ASCII file and later import it into a DB2 UDB for Windows 2000 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, refer to the *Command Reference*.

Moving Data from a Workstation to a S/390 or AS/400 Database Server

To export to an S/390 or AS/400 database server:

1. Export the rows of information from the DB2 UDB for Unix, Windows NT, Windows 2000, OS/2 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 Universal Database 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.

For more information on using these commands, refer to the *Command Reference*.

Chapter 10. Security

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

For more information on setting up security with DCE, refer to the *Administration Guide*, and the database and DCE manuals for your host or AS/400 database server.

Note: When using DB2 Connect with DCE security, DCE software is required to be installed at the DB2 client workstation and on the host or AS/400 database server, but it is not necessary to install it on the DB2 Connect workstation. For further information about software prerequisites for DCE, refer to your *DB2 Connect Quick Beginnings* book.

Authentication

As DB2 Connect administrator, in cooperation with your host or AS/400 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 host or AS/400 server
- Validation at the host or AS/400 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 or APPN nodes. For more information about updating these directories, see "Chapter 6. Updating Database Directories" on page 73.

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 server, if you will use `AUTHENTICATION=SERVER`, you must set up all the necessary userids and passwords on the server system.

2. If you use DCE Directory Services, authentication works differently. For more information, see “Security with DCE Directory Services” on page 206.

The following authentication types are allowed with DB2 Connect:

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.

SERVER_ENCRYPT
As for SERVER authentication, the user name and password are validated at the DB2 Connect workstation, but the transferred passwords are encrypted at the client and decrypted at the DB2 Connect workstation.

DCS The user name and password are validated at the host or AS/400 database server.

DCS_ENCRYPT
As for DCS authentication, the user name and password are validated at the host or AS/400 database server, but the transferred passwords are encrypted at the client and, depending on the authentication type specified at the DB2 Connect workstation, decrypted at the DB2 Connect workstation or host or AS/400 database server.

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

SERVER_ENCRYPT and DCS_ENCRYPT authentication have the same semantics as SERVER and DCS authentication in terms of authentication location. They differ in that any transferred passwords will be encrypted at the source (the client or the DB2 Connect server) and decrypted at the target (the DB2 Connect server or the host or AS/400 database server) as specified by the authentication type catalogued at the source.

Encrypted and non-encrypted values with matching authentication locations can then be used to choose different encryption combinations between client and DB2 Connect server or DB2 Connect server and host or AS/400 database server, while not affecting where authentication takes place. Here are some examples of how this might be used in a gateway scenario, where ″gateway″ is used to denote the DB2 Connect server:

Authentication at Client	Authentication at Gateway	Authentication Location	Client-Gateway Encryption?	Gateway-Server Encryption?
SERVER_ENCRYPT	SERVER	gateway	yes	no
DCS_ENCRYPT	DCS	server	yes	no
DCS	DCS_ENCRYPT	server	no	yes
DCS_ENCRYPT	DCS_ENCRYPT	server	yes	yes

The only APPC security parameter supported when either SERVER_ENCRYPT or DCS_ENCRYPT are used is SECURITY=NONE.

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. 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 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 to specify what security information will flow at the communications layer:

SAME Only the user name is passed to the host or AS/400 database server.

PROGRAM
The user name and password are passed to the host or AS/400 database server.

NONE
No security information flows.

Table 5 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 5. Valid Security Scenarios for APPC connections

Case	Authentication setting in the database directory entry at the DB2 Connect workstation	Security	Validation
1	CLIENT	SAME	Client
2	SERVER	SAME	DB2 Connect server
3	SERVER	PROGRAM	DB2 Connect server and host or AS/400 database server
4	SERVER_ENCRYPT or DCS_ENCRYPT	NONE	host or AS/400 database server
5	DCS	PROGRAM	host or AS/400 database server
6	DCE	NONE	DCE security server

If remote clients are connected to a DB2 Connect Enterprise Edition server, specify the following authentication and security types:

- If a remote client is connected to a DB2 Connect server 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 server is `CLIENT`, specify `CLIENT` at each remote client.
- If the authentication type at the DB2 Connect server is `SERVER`, `SERVER_ENCRYPT`, `DCS`, or `DCS_ENCRYPT`, specify one of these types at each remote client. (Which of these 4 types you specify at the remote client makes no difference.)

Notes:

1. For AIX systems, all login 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 host or AS/400 database 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 6. Valid Security Scenarios for TCP/IP connections

Case	Authentication setting in the database directory entry at the DB2 Connect workstation	Validation
1	CLIENT	Client
2	SERVER or SERVER_ENCRYPT	DB2 Connect workstation
3	Not applicable	None
4	DCS or DCS_ENCRYPT	host or AS/400 database 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 5 on page 114 and Table 6. 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 server.)

  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 server only. The password is sent across the network from the remote client to the DB2 Connect server but not to the host or AS/400 database server.
- In case 3, the user name and password are validated at both the DB2 Connect server and the host or AS/400 database 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 host or AS/400 database server.

Because validation is performed in two places, the same set of user names and passwords must be maintained at both the DB2 Connect server and the host or AS/400 database server.

- In case 4, the user name and password are validated at the host or AS/400 database server only. The user ID and password are sent across the network from the remote client to the DB2 Connect server and from the DB2 Connect server to the host or AS/400 database 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.

Changing Your MVS Password

DB2 Connect now provides an ability to change user passwords. This facility is especially useful for situations where host security service such as Resource Access Control Facility (RACF) is used to authenticate users. Previously changing host password would require users to log in to a TSO session to change their password. With the new password maintenance support provided by the DB2 Connect products users can issue SQL CONNECT statement from DB2 Command Line Processor (CLP), use PASSWORD button on DB2 Client Configuration Assistant (CCA), or press CHANGE button on the ODBC login dialog to change their host password.

Changing MVS passwords from DB2 Connect workstations connected to DB2 for OS/390 V5.1 via TCP/IP requires that the DB2 OS/390 Extended Security Field be set to ″Yes″. This field appears in the DB2 OS/390 DSNTIPR panel.

Changing of MVS passwords on host systems connected via SNA requires that a special password expiration management program be set up on the host and that DB2 Connect workstation be configured to communicate with this host program.

The host password expiration management program is provided as part of the following MVS program products:

- MVS/ESA SP Version 4.2 or higher (password expiration management is a part of the APPC/MVS component)
- CICS/ESA Version 3.3 or higher

and has IBM Resource Access Control Facility (RACF) 1.9.2 installed.

You need to:

1. Configure the host's transaction program to receive your requests for password expiration maintenance.
2. Configure your DB2 Connect workstation for communications with the host transaction program.

Configuring the DB2 Connect Workstation for Password Expiration Management

Once the host password expiration management transaction program is configured, you will need to configure your DB2 Connect workstation to communicate with the host program. This configuration involves two steps:

1. Define a symbolic destination name for the host password maintenance program in the SNA subsystem on your DB2 Connect workstation.
2. Record this symbolic destination name in the DCS directory for the databases that reside on this host system.

Step 1. Define Symbolic Destination Name

How you define symbolic destination name for the host password expiration management program depends on what SNA subsystem you are using:

- If you are using an SNA subsystem that can be configured by the DB2 Client Configuration Assistant (CCA), then you should use the CCA to configure this symbolic destination name. You will need to obtain the LU name for the password expiration management program from your MVS administrator.
- If your SNA subsystem can not be configured using the CCA, you should follow documentation provided with your SNA subsystem to configure the symbolic destination name. You will need to obtain the following information from your MVS administrator:
  - Network name for the host that you are connecting to.
  - LU name for the host password expiration management program.

  When configuring symbolic destination name, you will also need to specify x'06F3F0F1' (hexadecimal number) for the Transaction Program (TP) name and set security to `NONE`. You can specify mode such as #INTER or any other mode that your MVS may suggest to you.

Step 2. Record Symbolic Destination Name in the DCS Directory

If you are running DB2 Connect on a platform that provides the CCA, then you should use it to update your DCS directory with the symbolic destination name for the host password expiration management program. You should be able to do this regardless of the SNA subsystem on your DB2 Connect workstation.

You can also use **catalog dcs database** command (from DB2 CLP) to record the symbolic destination name in the DCS directory. For example:

```
catalog dcs database db1 as dsn_db_1 parms ",,,,,,,CHGPWD_SDN=pempgm"
```

records *pempgm* as the symbolic destination name that is to be used when users request to change passwords for database *db1*.

Configuring the Host for Password Expiration Management

For more detailed information on MVS passwords, consult one of the following online publications:

APPC/MVS:
http://www.s390.ibm.com/products/appc/library

SecureWay Communications:
http://www.ibm.com/software/network/commserver/library

TxSeries:
http://www.ibm.com/software/ts/txseries/library

Additional Hints and Tips About Security

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

Extended Security Codes

Until DB2 Universal Database 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 Universal Database for OS/390 Version 5.1 introduced 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.

To exploit this, the DB2 Universal Database for OS/390 `ZPARM` installation parameter for extended security should be set to the value `YES`. Use the DB2 Universal Database 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`. In the case of an expired password, PC, UNIX, Apple Macintosh, and Web applications using DB2 Connect will receive error message SQL01404.

TCP/IP Security Already Verified

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

Desktop ODBC and Java Application Security

Workstation ODBC and Java applications use dynamic SQL. This may create security concerns in some installations. DB2 Universal Database 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. Refer to the *Command Reference* to see how DYNAMICRULES can be specified through DB2 Connect.

DB2 Universal Database and DB2 Connect provide 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 *Installation and Configuration Supplement* for further information.

Password Change Support

If an SQL CONNECT statement returns a message indicating that the user ID's password has expired, with DB2 Connect Version 5.2 and later it is possible to change the password without signing on to TSO. Through DRDA, DB2 Universal Database for OS/390 can change the password for you.

The old password along with the new password and the verify password must be supplied by the user. If the security specified at the DB2 Connect Enterprise Edition server is `DCS` then a request to change the password is sent to the DB2 Universal Database for OS/390 database server. If the security specified is `SERVER` then the password on the DB2 Connect server is changed.

An additional benefit is that a separateLU definition is not required. Refer to the DB2 Connect Enterprise Edition *Quick Beginnings* manual for additional information.

Chapter 11. 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 host or AS/400 database server to the appropriate DB2 Universal Database SQLCODEs.

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 6. Updating Database Directories" on page 73. For information about using DCE, see "Appendix D. Using DCE Directory Services" on page 199.

If you port an application directly from a host or AS/400 database server (such as DB2 UDB 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 host or AS/400 database server to the appropriate DB2 UDB SQLCODEs. The following files are copies of the default SQLCODE mapping:

dcs1dsn.map
: Maps DB2 for MVS/ESA and DB2 UDB for OS/390 SQLCODEs

dcs1ari.map
: Maps DB2 for VSE & VM SQLCODEs

dcs1qsq.map
: Maps DB2 UDB 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 host or AS/400 database server that does not have SQLCODE mapping (a non-IBM database 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 6. Updating Database Directories" on page 73. For information about using DCE, see "Appendix D. Using DCE Directory Services" on page 199.

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 represents one of the following:

sqlcode The SQLCODE from the host or AS/400 database server.

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 host or AS/400 database server. *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

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 represents the output SQLCODE. 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 host or AS/400 database server products support only CHAR tokens. The character **i** must be lowercase.

n A number or numbers indicating which host or AS/400 database server tokens are used. They are arranged in the order desired for placement in the output SQLCA. The number indicates the host or AS/400 database server token; the arrangement indicates the order in which the tokens will be placed in the SQLCA.

For example, the host or AS/400 database 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 6 on page 125 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 6. An SQLCODE Mapping File

Each mapping statement in the file is described as follows:

1. The SQLCODE is mapped from -007 to -007. The first input token received from the host or AS/400 database 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 host or AS/400 database 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 host or AS/400 database 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 host or AS/400 database 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 host or AS/400 database 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 12. 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.

Performance Concepts and Tools

Performance is the way a computer system behaves given a particular workload. It is affected by the available resources 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 host or AS/400 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 7 on page 128 shows the path of data flowing between the host or AS/400 database server and the workstation through DB2 Connect.

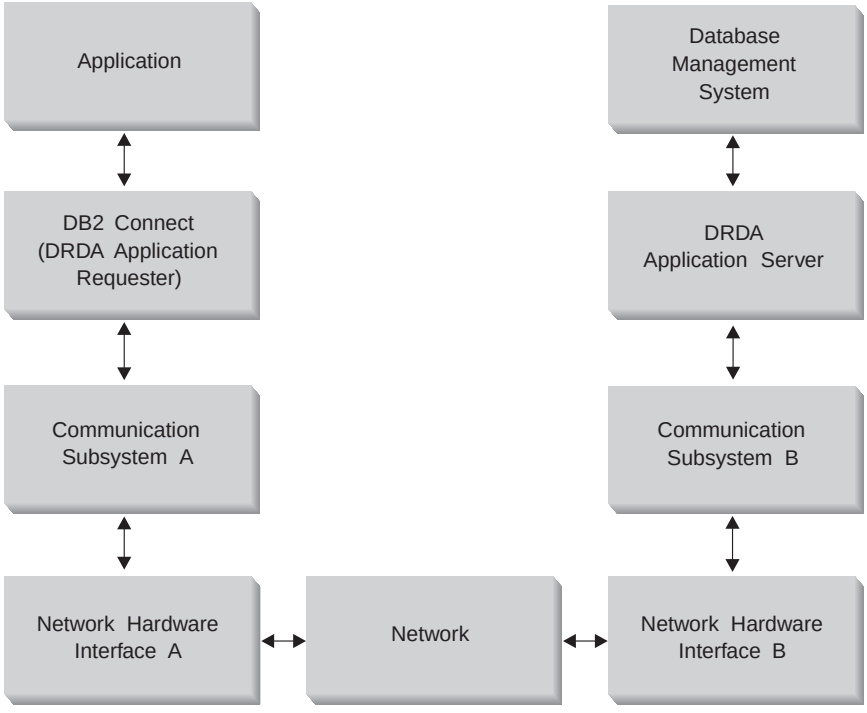


Figure 7. Data Flows in DB2 Connect

- The host or AS/400 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 SNAplus2 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 off-load 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 130.

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:

- 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 7. 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 and Windows 2000	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 and Windows 2000	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.

Table 7. Performance Tools (continued)

System	Tool	Description.
OS/2	DatagLANce	A trace tool that presents performance-related data to users graphically.
UNIX-based	netstat	Handles TCP/IP traffic.

Optimizing ODBC Access

DB2 Universal Database provides special optimization designed to improve communication performance through ODBC. These enhancements are available to Microsoft Access, Lotus Approach, or Visual Basic. You can gain the benefit of faster ODBC throughput using DB2's Client Configuration Assistant (CCA).

To activate the optimized ODBC, perform the following:

If you are defining a new connection:

1. Start the DB2 CCA.
2. Select the database alias that you want to optimize.
3. Click the **Properties** push button.
4. Ensure that the **Register this Database for ODBC** check box is selected.
5. Select the radio button that describes how you would like to register this database.
6. Click the **Settings** push button.
7. Click the **Optimize** push button and select the application whose OBDC settings you wish to optimize.
8. Click **OK** and exit the CCA.

If you are updating an existing connection:

1. Start the DB2 CCA.
2. Select the database alias that you want to optimize.
3. Click the **Properties** push button.
4. Click the **Settings** push button.
5. Click the **Optimize** push button from the CLI/ODBC Settings Window and select the application for which you want to optimize.
6. Click **OK** and exit the CCA.

For more information about the CCA, to your *DB2 Connect Quick Beginnings* book. You can find the most recent information about optimizing ODBC

access to DB2 for OS/390 by pointing your browser to:
http://www.ibm.com/software/data/db2/os390/odbcatlg.html

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 you require programming logic 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
```

For more details, refer to the *SQL Reference*.

For information about using compound SQL in an application, see "NOT ATOMIC Compound SQL" on page 52. For information about using compound SQL with the import utility, see "Using Import and Export Utilities" on page 108.

Stored procedures help to reduce network traffic by placing program logic at the server. In DB2 prior to Version 5.0, a stored procedure could return output

parameters only, and a separate commit command had to be issued by the application. This resulted in two network trips. In DB2 Versions 5.0 and above, you can commit automatically when exiting the procedure. You can also return results sets, which minimize application logic at the client.

For information about using stored procedures, see "Stored Procedures" on page 50.

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 use of the network bandwidth and reduces the CPU overhead of both the host or AS/400 database 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, refer to 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 136.

To invoke blocking, use the BLOCKING option of the prep or bind command. (For more information, see "The BIND Command" on page 92.) 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 *Application Development 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.
- ODBC applications use dynamic SQL. You may use the CLI/ODBC static profiling feature to improve performance. This feature allows you to capture and convert ODBC calls into static statements stored in a database package. The actual performance you will get depends on the complexity of your application. For more information, refer to *CLI Guide and Reference*.

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, refer to 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, refer to 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 130.

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. These settings 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 host or AS/400 database server 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 host or AS/400 database server 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 Charge-Back Accounting on DB2 Universal Database for OS/390” on page 54.

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 11. SQLCODE Mapping” on page 121.

Connection Pooling

DB2 Connect Enterprise Edition servers often provide database connections for thousands of simultaneous client requests. Establishing and severing connections to the database server can be a very resource intensive process that adversely affects both database server and DB2 Connect server performance. This is especially evident in web environments where each visit to a web page can require building a new connection to the database server,

performing a query and terminating a connection. To reduce this overhead, DB2 Connect Enterprise Edition uses *connection pooling* to maintain open connections to the database in a readily accessible pool.

How Connection Pooling Works

Connection pooling is transparent to applications connecting to the host through DB2 Connect. When an application requests disconnection from the host, DB2 Connect drops the inbound connection with the application, but keeps the outbound connection to the host in a pool. When a new application requests a connection, the DB2 Connect uses one from the existing pool. Using the already-present connection reduces the overall connection time, as well as the high CPU connect cost on the host.

To use connection pooling, the following APAR must be applied to DB2 for OS/390 Version 6.1:

```
APAR PQ33473
```

DB2 Connect agents can be in one two states: idle or active. An agent is active when it is executing work for an application. Once this work is completed the agent goes into an idle state awaiting further work from the same or a different application. All idle agents are kept together in what is known as the idle agent pool. You can configure the size of this pool using the NUM_POOLAGENTS configuration parameter. This parameter equals the maximum number of idle agents you wish the system to maintain. Setting this parameter to zero is equivalent to turning off the connection pooling feature.

DB2 Connect does not establish connections to the database before receiving its first client request. If you wish, however, you may fill the pool of idle agents before any clients make a request. The pool can be filled on start-up using the NUM_INITAGENTS configuration parameter. This parameter determines how many idle agents should be created at start up time. These idle agents will not initially have connections to the host database server.

When a client requests a connection to the host, DB2 Connect will attempt to get an agent from among those in the pool that have a connection to the host database server. If that fails, it will try to find an available agent in the idle pool. If the pool is empty, DB2 Connect will create a new agent.

You can control the maximum number of agents that can be concurrently active using the MAX_COORDAGENTS configuration parameter. Once this number is exceeded, new connections will fail with error sqlcode SQL1226. (This code means that the maximum number of concurrent outbound connections has been exceeded.)

The db2 registry variable DB2CONNECT_IN_APP_PROCESS allows applications running on the same machine as DB2 Connect EE to either have

DB2 Connect run within the applications process, default behavior, or to have the application connect to the DB2 Connect EE Server and then have the host connection run within an agent. In order for an application to use connection pooling the connections to the host must be made from within the DB2 Connect EE Server agents and thus DB2CONNECT_IN_APP_PROCESS must be set to NO.

DB2 Connect Connection Concentrator

DB2 Connect's *connection concentrator* technology allows DB2 Connect Enterprise Edition servers to provide support to thousands of users simultaneously executing business transactions, while drastically reducing resources required on the S/390 host or AS/400 database servers. It accomplishes this goal by concentrating the workload from all applications in a much smaller number of S/390 host or AS/400 database server connections. While this may seem similar to the connection pooling function described above it is in fact a more sophisticated approach to reducing resource consumption for very high volume OLTP (On-line Transaction Processing) applications.

Connection pooling saves the cost of establishing a connection when one is no longer needed by a terminating application. In other words, one application has to disconnect before another one can reuse a pooled connection.

Connection concentrator, on the other hand, allows DB2 Connect to make a connection available to an application as soon as another application has finished a transaction and does not require that other application to disconnect. In essence, a database server connection and its associated host and DB2 Connect resources are used by an application only while it has an active transaction. As soon as the transaction completes, the connection and associated resources are available for use by any other application that needs to have a transaction executed.

How the Connection Concentrator is Implemented

In previous versions of DB2 Connect, every active application had an Engine Dispatchable Unit (EDU) which managed the database connection as well as any application requests. This EDU was typically referred to as the *coordinator agent*. Each coordinator agent tracked the state, or context of the application and EDU. Each EDU takes a significant amount of memory when the number of connections increase, and context switching between agents results in additional overhead.

In the above architecture, there is a one-to-one relationship between connections and EDUs. The connection concentrator, however, permits a many-to-one relationship between connections and EDUs. That is, the relationship of connections (X) to EDUs (Y) is now X >= Y.

The connection concentrator splits the agent into two entities, a *logical agent* and a *worker agent*. Logical agents represent an application, but without reference to a particular EDU. The logical agent contains all the information and control blocks required by an application. If there are *n* applications connected to the server, there will be *n* logical agents on the server. Worker agents are physical EDUs that execute application requests, but which have no permanent attachment to any given application. Worker agents associate with logical agents to perform transactions, and at transaction boundary end the association and return to the available pool.

An entity known as the *logical agent scheduler* assigns worker agents to logical agents. Limitations in the number of open file handles on certain computing platforms may result in more than one scheduler instance when the number of logical agents exceeds the file handle limit.

Activating the Concentrator

To use the connection concentrator, the following APAR must be applied to DB2 for OS/390 Version 6.1:

```
APAR PQ33473
```

The database manager configuration parameter MAX_LOGICAGENTS sets the maximum number of logical agents. You can activate the concentrator feature by setting the value of MAX_LOGICAGENTS to any number greater than the default. The default value for MAX_LOGICAGENTS is equivalent to the value of MAX_COORDAGENTS. Because each application will have one logical agent, MAX_LOGICAGENTS actually controls the number of applications that can be connected to the database instance, while MAX_COORDAGENTS controls the number of inbound connections that can be active at any time. MAX_LOGICAGENTS will take a numeric range from MAX_COORDAGENTS up to 64,000. The default number of logical agents is equal to MAX_COORDAGENTS.

Several existing configuration parameters are used to configure agents. These parameters are as follows:

MAXAGENTS
: Maximum number of worker agents.

MAX_COORDAGENTS
: Maximum number of active coordinator agents.

NUM_POOLAGENTS
: Agents pool size. The agent pool includes inactive agents and idle agents.

NUM_INITAGENTS
: Initial number of worker agents in the pool. These will be idle agents.

XA Transaction Support

The architecture of the connection concentrator allows DB2 Connect to provide tightly coupled XA transaction support to DB2 for OS/390 and DB2 for AS/400. The concentrator will associate a worker agent with a particular XA transaction (single XID) as it would for any other transaction. However, if the XA transaction is ended by `xa_end()` (branch boundary), the worker agent will not release itself into the general pool. Instead, the worker remains associated with that particular XA transaction. When another application joins the same XA transaction, the worker agent will be attached to that application.

Any transaction boundary call will return the agent to the pool. For instance, `xa_prepare()` with read only, `xa_rollback()`, `xa_recover()`, `xa_forget()`, `xa_commit()`, or any XA error that causes rollback will return the agent to the normal pool. `Xa_end()` itself only ends the transaction branch, and this is not sufficient to end its association with the XID.

Examples

1. Consider an environment where 4,000 or more concurrent connections are needed. A web server that uses CGI applications, or an office system with many desktop users can both exceed this requirement. In these cases, efficiency will usually require that DB2 Connect operate as a stand-alone gateway; that is, the database and the DB2 Connect system are on separate machines.

   The DB2 Connect server system may not be able to maintain 4,000 simultaneous open connections to the database machine. In most cases, the number of transactions occurring at any given moment will be considerably less than the number of concurrent connections. The system administrator could then maximize the efficiency of the system by setting the database configuration parameters as follows:

   ```
   MAX_LOGICAGENTS =  4,000
   MAX_AGENTS      =  1,000
   MAX_COORDAGENTS =  1,000
   NUM_POOLAGENTS  =  1,000
   ```

   The concentrator will keep open up to 4,000 concurrent sessions, even though the gateway is only managing 1,000 transactions at a time.

2. In the above example, worker agents will constantly form and break associations to logical agents. Those agents that are not idle may maintain a connection to the database but are not participating in any particular transaction, hence they are available to any logical agent (application) that requests a connection.

   The case of XA transactions is somewhat different. For this example, we may assume that a TP Monitor is being used with a DB2 Connect gateway and an OS/390 or AS/400 database. When an application requests a connection, the concentrator will either turn an inactive agent over to serve that request, or create a new worker agent. Let us assume that the

application requests an XA transaction. An XID is created for this transaction and the worker agent is associated with it.

When the application's request has been serviced, it issues an `xa_end()` and detaches from the worker agent. The worker agent remains associated with the XID of the transaction. It can now only service requests for transactions with its associated XID.

At this time, another application may make a request for a non-XA transaction. Even if there are no other available worker agents, the agent associated with the XID will not be made available to the second application. It is considered active. The second application will have a new worker agent created for it. When that second application completes its transaction, its worker agent is released into the available pool.

Meanwhile, other applications requesting the transaction associated with the first agent's XID may attach and detach from that agent, which executes its dedicated XA transaction for them. Any application requesting that particular transaction will be sent to this worker agent if it is free.

The worker agent will not be released back into the general pool until an application issues a transaction boundary call (not `xa_end()`). For instance, an application might end the transaction with `xa_commit()`, at which point the worker agent drops its association with the XID and returns to the available pool. At this point any requesting application can use it for either another XA, or a non-XA, transaction.

Restrictions

There are a number of important restrictions to the use of the gateway concentrator. Review the following information in its entirety before attempting to use the connection concentrator on your system.

- The connection concentrator can only be used by DB2 Version 7 or later clients.
- Only DB2 for OS/390 or DB2 for AS/400 hosts are supported by the concentrator.
- The concentrator relies on the TCP/IP protocol to establish inbound connections from local and remote clients. Only inbound connections using TCP/IP or Local (IPC) will be able to take advantage of pooled outbound connections. The concentrator will accept connections via other communications protocols such as SNA, but you will not be able to use its XA concentration features with that connection.
- You should not use the static `SET` statement in your client applications if the concentrator is enabled on the gateway. DB2 will not return an error if you use static `SET`, but your application and any other applications that share the same outbound connection could be adversely affected.
- For `SET` statements, only immediate execution is supported.

- If you declare global temp tables, they must be closed explicitly at transaction or branch boundary. Failure to close the tables may result in an error during later transactions.
- For XA tightly coupled transaction support, all applications that participate in the same XA transaction must use the same gateway to connect to the host.
- Only applications that close any withhold cursors transaction boundary can benefit from the concentrator. Transactions that do not close withhold cursors will still go through, but will be assigned a dedicated worker agent and hence will not be able to use the concentrator's full feature set.
- All applications participating in the same XA transaction must have the same CCSID and use the same userid to make the connection.
- If an outbound connection was established to support two-phase connection, that connection's agent can only be used to support two phase connections. Similarly, agents established to support a one-phase connection can only support one phase connections.
- Authentication type `DCS_ENCRYPT` will not work with the concentrator in Version 7.
- The concentrator only supports dynamic SQL from the command line interface. Dynamic prepare requests from embedded dynamic SQL applications will be rejected. Your applications should be altered so as to either use static SQL or to use the CLI for dynamic SQL statements.

Database Tuning

System performance will be affected by the performance of the host or AS/400 database 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 host or AS/400 database server system performance documentation for more information.

For DB2 Universal Database for AS/400, you may be able to improve performance by using the uncommitted read (UR) or no commit (NC) bind options to avoid journalling.

Note: 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.

Tuning DB2 for OS/390

OS/390 V1R3 is the minimum requirement for TCP/IP support. OS/390 V2R5 or later is highly recommended.

The Distributed Data Facility (DDF) is responsible for connecting distributed applications to DB2 for OS/390. The DDF should be set up as an application server. To do this, you can either insert the LU name of the remote system into the SYSIBM.LUNAMES table, or insert the LUNAME, SYSMODENAME, USERSECURITY, ENCRYPTPSWDS, MODESELECT, and USERNAMES values into the SYSIBM.SYSLUNAME table. Then perform a DDF update to the Boot Strap Data Set (BSDS). For example:

```
DDF LOCATION=LOC1,LUNAME=LU1,PORT=8000,RESPORT=8001
```

For best performance, you should use the recommended DDF address space prioritisation (slightly lower or equal to DBM1 if you are in COMPAT mode). Use RACF caching of authorizations in VLF, and use V5 package authorizations caching if you can. A value of CACHEPAC=32768 is sufficient for most operations.

Since DDF will try to connect to VTAM, VTAM must be active when DDF starts. A sample VTAM APPL definition is included below:

```
SYD51TC* APPL  AUTH=(ACQ),                              X
           PARSESS=YES,                                 X
           HAVAIL=YES,                                  X
           EAS=1600,                                    X
           APPC=YES,                                    X
           DSESLIM=1024,                                X
           DMINWNL=512,                                 X
           DMINWNR=512,                                 X
           AUTOSES=1,                                   X
           SECACPT=ALREADYV,                            X
           SRBEXIT=YES,                                 X
           SYNCLVL=SYNCPT,                              X
           MODETAB=DB2MODET,                            X
           VPACING=63                                   X
```

You can optimize inactive thread processing in OS/390. In V3, you are allowed up to 10,000 concurrently connected clients, and up to 25,000 in V4 and V5. In all cases, the maximum number that can be concurrently active, however, is 1999. Each workstation client can stay connected when it is inactive; its thread is placed on an inactive chain at each commit.

The DSNZPARM parameters CMTSTAT, CONDBAT and MAXDBAT affect thread processing. For best performance, set CMTSTAT to INACTIVE, adjust CONDBAT to

the maximum number of connected DBATs that provide good performance, and MAXDBAT to the maximum acceptable number of active DBATs.

For a complete discussion on connecting DB2 for OS/390 in a DRDA network, including VTAM configuration, refer to the *Connectivity Supplement*.

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, Windows NT or Windows 2000)
- 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 8 shows when conversion takes place.

Table 8. 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		

Table 8. Data Conversion (continued)

	Intel	IEEE	S/370 & S/390	OS/400
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 host or AS/400 database 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.

Network Tuning

The best way to improve overall performance in a distributed database environment is to eliminate delays from the network. It is common for network administrators to consider a network to be more efficient if it collects as much data as possible between transmissions. This approach doesn't work for applications such as distributed databases because it builds delays into the network. The end-user doesn't see the efficiency of the network, only the delays.

Most network devices have delay parameters, and most of them default to values that are very bad for distributed databases. To improve performance you should locate these parameters and if possible, set them to zero. In addition you should ensure that the buffer size on the device is large enough to prevent retransmits due to lost data. For instance, UNIX systems typically

have a Transmit or Receive queue depth default of 32. For better results, set the queue depth to 150. A corresponding parameter on DLC settings is the Receive Depth, which should also be 150.

The IOBUF parameter is set too low at most sites. It is usually set at 500, but experience has shown that a value of 3992 works best if you are moving large amounts of data, especially for channel connections such as ESCON or 3172.

For SNA connections, you should set the Mode Profile of any workstation software to 63. In general, receive pacing values throughout the network should be set to their highest value, so the VPACING and PACING parameters on the DB2 APPL statement, and the PU/LU for the workstation in a switched major mode should also be set to 63. What this will do is allow the amount of message flows before the sender must wait for a response to increase progressively.

On a LAN system the DLC or LLC transmit and receive window sizes can have a dramatic effect on performance. The send value should be set to seven or more, and for most configurations a receive value of four or less works best.

If you are running Ethernet, you should set the TCP segment size to 1500 bytes. On a token ring or FDDI network this value should be 4400 bytes, and if you are using an ESCON adapter with TCP/IP, the segment size should always be 4096.

Finally, for TCP/IP networks, the TCP Send and Receive buffer sizes should be set higher than 32768. A value of 65536 is generally best.

Note: Establishing a connection from the gateway to the server (outbound connection) is much more expensive than establishing a connection from a client to the gateway (inbound connection). In an environment where thousands of clients frequently connect to and disconnect from the server through the gateway, a substantial amount of processing time is spent establishing outbound connections. DB2 Connect provides connection pooling over TCP/IP. When a client requests disconnection from the server, the gateway drops the inbound connection with the client, but keeps the outbound connection to the server in a pool. When a new client comes into the gateway to request a connection, the gateway provides an existing one from the pool thus reducing the overall connection time and saving the high CPU connect cost on the server.

For more information about connection pooling under DB2, refer to the *Administration Guide*.

A summary of network performance tuning methods is provided in the following table.

What to Look For	Example	Setting	Notes
Deliberate Delays	Delay parameters on network devices	Set to 0.	Defaults are usually higher.
Buffers	IOBUF parameter	Set up to 3992.	Particularly useful for ESCON or other channel adapter.
	RUSIZE	Optimum size is 4096.	Setting RUSIZE and RQRIOBLK to same size may give best performance.
	Pacing	VPACING, PACING, and Mode Profiles should be set to 63.	Use adaptive pacing where applicable.
Adapter Settings	Transmit/Receive queue depth	Recommended value is 150.	Default is usually 32.
	DLC Windowing on SNA	Set transmit window size high (>7). Set receive window size low (for example, at 1), test and increment repeatedly to find ideal value.	Every logical device adds delays. Simply network topology as much as possible.
TCP Settings	Segment Sizes	1500 on Ethernet, 4400 on token ring and FDDI.	ESCON adapters used for TCP/IP should always be set to 4096.
	Send/Receive Space Sizes	Should be 64K for both.	Default is only 8192 for Windows. Can be set in the Windows registry.

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 host or AS/400 database 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 host or AS/400 servers, 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 host or AS/400 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 the host or AS/400 server, 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.

Additional SNA Performance Tuning Hints and Tips

This section contains additional SNA performance tuning hints and tips for use with DB2 Connect.

General Performance Information for DB2 Connect

The performance characteristics of DB2 Connect are that it predominantly uses the processor and performs very little I/O. In general, the faster the processor speed, the faster DB2 Connect will run. DB2 Connect fully exploits SMP processor configurations.

A fast DB2 Connect Enterprise Edition server can handle an SQL request/reply pair in less than five milliseconds, not counting client time, network time, and processing time at the host or AS/400 server. A simple SQL statement or query with a few rows of data could be completed end-to-end in less than 0.1 seconds (from client to the host or AS/400 server and back).

When there are more than four or five SQL statements in a query, then the use of stored procedures will help to ensure high OLTP performance and to avoid increases in lock contention due to network delays between SQL statements.

Performance problems are usually caused by to the type of host attachment in use, network routing and tuning characteristics, and application design. Some general DB2 Connect performance information can be found in “Other DB2 Connect Performance Information Sources” on page 152.

Selection and Tuning of the Network Attachment

In order of likely best performance when using DB2 Connect, various types of network attachment include:

1. Channel attachment card
2. IBM 3172 Model 3, or newer models, or equivalent
3. IBM 2216
4. Open System Adaptor Card (OSA-2, not OSA-1)
5. IBM 3745 with Network Control Program (NCP)
6. IBM 3174 Terminal Controllers, or equivalent

The last option is not recommended - see below.

The recommended best way to connect to the host is to use ESCON channel attachment cards for AIX, Windows NT or Windows 2000. The IBM 3172 Model 3 and 2216 also perform well, but they tend to deliver throughput inferior to ESCON.

When using AIX with ESCON cards, please apply the PTFs related to MPC (Multi Path Channel). Without these PTFs the AIX SNA ESCON driver may deliver worse performance. See "Multi Path Channel Support for SNA over ESCON" on page 153 for more details. Further information can also be found at: http://www.networking.ibm.com.cms/cmsnew01.html

See "How to Tune DB2 Connect Connections via NCP" on page 154 for a checklist of which Communications Server, NCP, and VTAM parameters to tune to optimize DB2 Connect performance. All the non-NCP specific recommendations are applicable to all types of DB2 Connect and client/server attachments.

The OSA-2 card on System/390 might not deliver throughput as high as a 3272 Model 3 when there is a high volume of small transactions, owing to its lower frames-per-second capability. See "Information about OSA-2 Enhancements" on page 156 for details of some recent enhancements.

3145 with NCP is usually tuned specifically for existing network traffic. Consequently it might not perform as well for database client/server applications. Most DB2 Connect performance problems are due to the time delay between the NCP and VTAM and/or between NCPs. See "How to Tune DB2 Connect Connections via NCP" on page 154, which provides a tuning checklist.

In general, we recommend avoiding the use of 3174 Terminal Controllers because their packet size (RU size) of 256 bytes is too small. 3174 microcode level C is required in order to provide Independent LU support for APPC database connections. Some OEM 3174 equivalents may have similar dependencies.

Other DB2 Connect Performance Information Sources

- Search the DB2 Technical Library web site at http://www.ibm.com/software/data/db2/library. Search the DB2 Universal Database library for ″Technotes″ with the keywords ″DB2CONNECT″ and ″Performance″ to find the latest information with a section on DB2 Connect considerations on the World Wide Web.
- There are also a number of DB2 client/server performance reports involving DB2 Connect in http://www.ibm.com/software/data/performance

Multi Path Channel Support for SNA over ESCON

Multi Path Channel (MPC) support for SNA over ESCON allows a system running IBM eNetwork Communications Server to use an ESCON adapter to create an MPC linkstation to the host. MPC is typically faster than CDLC because:

1. MPC uses separate subchannels for read and write
2. MPC is not limited by IOBUF size. Frames are 4K and may be blocked together.

Tests have shown as much as a threefold improvement on an MPC link compared to an ESCON Channel Data Link Control (CDLC) link with an IOBUF size less than 1K. AIX SNA MPC requires ESCON and MVS VTAM V4R4 or later and feature code 4024 of Communications Server for AIX (5765-652). Windows NT systems must use IBM eNetwork Communications Server for Windows NT Version 6.

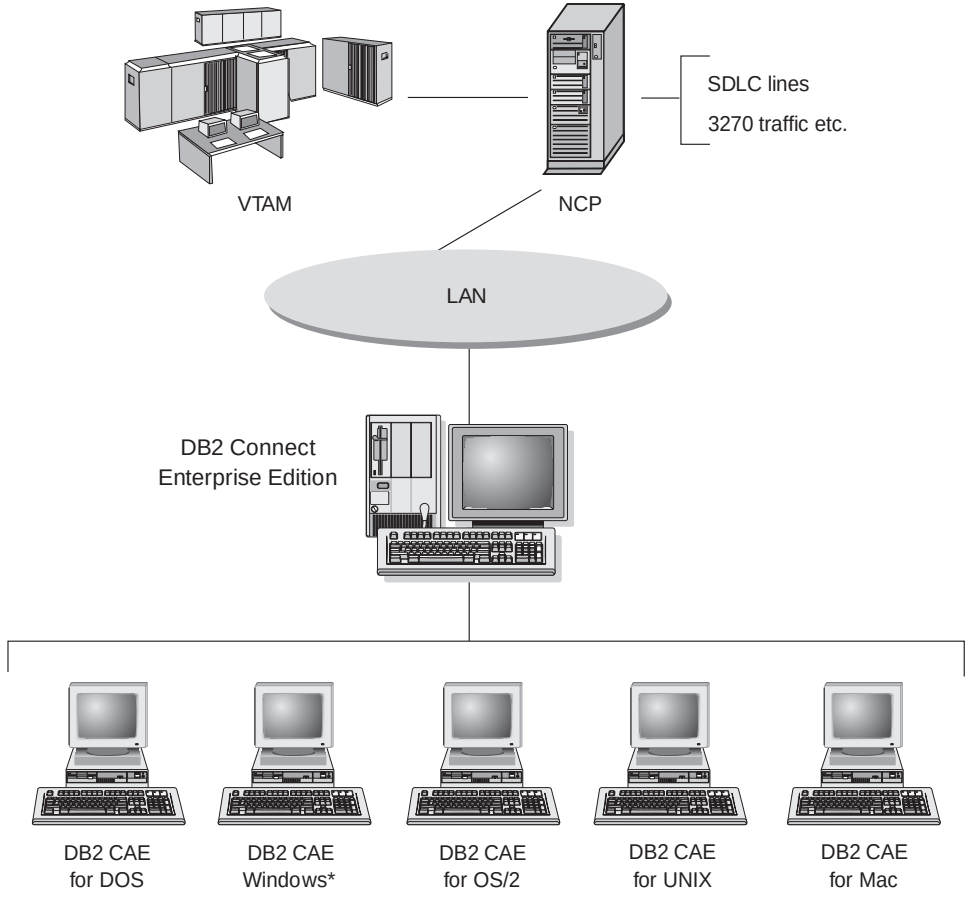
The following are the Communications Server for AIX PTFs required for MPC:

```
APAR #     PTF #    LPP name

IX67032   U449693   sna.books.chdoc
IX67032   U449693   sna.books.escdoc
IX67032   U449300   sna.rte
IX67032   U450027   sna.msg.en_US.rte
IX65820   U447759   sna.dlcchannel
IX67618   U449691   mpc.rte
IX65813   U447758   devices.mca.8fc3.rte
```

How to Tune DB2 Connect Connections via NCP

A typical network configuration might be:



*16 and 32-bit Windows operating systems.

Figure 8. DB2 Connect Enterprise Edition gateway SNA network scenario

This scenario focuses on the throughput and response time between the host or AS/400 database server to the DB2 Connect Enterprise Edition gateway and various parameters that could affect this.

Tuning Criteria

The suggested order in which to make these changes is:

```
1 - DELAY on PCCU macro*
2 - DLC/LLC Tuning*
3 - PIU size*
4 - Pacing window changes*
```

```
5 - DELAY on LINE macro*
6 - MAXBFRU changes
7 - LAN Frame sizes

* Major improvement in throughput is possible
```

PIU size (RU + 29 bytes): The RU size at the host and the DB2 Connect server should be maximized. This implies that the RU size should be large enough to contain the API crossing (both SEND and RECEIVE data for the transaction where possible) in order to minimize the number of times VTAM program stack must be traversed. Also, the network frame size may limit the maximum RU size if RU segmentation is not desired.

It is a good idea to set the DB2 Connect block size (RQRIOBLK), RU and pacing values such that RU * pacing >= RQRIOBLK. For example, the default RQRIOBLK size of 32K is a good value for most situations, and to exploit this you would set RU = 4K and receive window pacing to 8.

- RU size and pacing are set by the mode table which is defined on both DB2 Connect workstation and in VTAM. The mode table definitions should be the same in both places.
- RQRIOBLK is set using the DB2 UPDATE DBM CFG command.
- Network frame size I-frame is set in DLC configuration on the DB2 Connect workstation and in NCP.

Pacing window sizes: The session and VR pacing windows should be maximized: the largest value that does not cause network congestion or VR-held conditions, and so on, should be used. For a test environment set pacing to 0 (no pacing) or set it to the maximum value X'3F'.

Coat-tailing values (DELAY): Coat-tailing is controlled by the DELAY parameter. The DELAY Parameter in the PCCU macro controls outbound coat-tailing (outbound with reference to the host). The DELAY value in the LINE definition statement for the NCP controls inbound coat-tailing (inbound with reference to the host).

The DELAY value determines how long a PIU is held in the queue (NCP or VTAM) before it is transmitted. The purpose of this wait is to increase the possibility that other PIUs will arrive in the interim and all of these can be transmitted on a single channel program. For the lowest latency, the DELAY value should be set to 0. Changing the value of the outbound coat-tailing delay value to 0 should have no noticeable effect on the host except for improved performance for outbound traffic. Some improvement in inbound traffic performance will also be realized.

Changing the DELAY on the NCP to 0 should be done with a little more care. The value can be set to 0 if the NCP is not overloaded and the inbound traffic

does not consist of a significant percentage of small frames. Setting the values of DELAY to 0 may improve response time significantly, especially under light loads or test/benchmark environments.

```
VTAMB7   PCCU  CUADDR=CAF,
                                    AUTODMP=NO,
                                    AUTOIPL=NO,
                                    AUTOSYN=YES,
                                    BACKUP=YES,
                                    DELAY=0,
                                    VFYLM=YES,
                                    CHANCON=UNCOND,
                                    MAXDATA=32768,
                                    DUMPDS=NCPDUMP,
                                    OWNER=HOSTB7,
                                    SUBAREA=17


LNCTLS        GROUP   LNCTL=CA,CA=TYPE6,DELAY=0.0,TIMEOUT=500.0
CA0                 LINE         ADDRESS=00
PUCHAN0    PU            PUTYPE=5,TGN=1
CA1                 LINE         ADDRESS=01
PUCHAN1    PU            PUTYPE=5,TGN=1
```

DELAY considerations are documented in the *VTAM Network Implementation Guide*.

MAXBFRU: The MAXBFRU value should be set to a value two or three times larger than the largest PIU size.

DLC/LLC layer Tuning: Ensure that the LLC2 window sizes (DLC send and receive window counts) between the NCP and the DB2 Connect Enterprise Edition gateway are the same. This has a significant effect specially when the server is DB2 Connect for AIX. It is recommended that the send window count be set higher than the receive window count.

In general, for any SNA connection across a Token-ring the LLC2 timers/windows should be optimized. In some cases, this change led to a six-fold improvement in throughput and response time.

LAN Frame sizes: The token ring maximum frame size should be as large as possible.

Information about OSA-2 Enhancements

The following information is reproduced from the IBM WSC Flash document number 9718.

```
TITLE: WSC  FLASH 9718:  OSA-2 ENHANCEMENTS AVAILABLE
DOCUMENT ID G023691 UNCLASSIFIED

 Open Systems Adapter 2 (OSA-2) Systems Network Architecture (SNA)
```

enhancements are being made available earlier than previously announced. The enhancements are:

o SNA/APPN enhancements for OS/390, MVS/ESA, VM/ESA, and VSE/ESA

  - Enhanced availability: load balancing, redundancy, and overflow
  - Enhanced connectivity: increased Physical Unit (PU) support (from 255 PUs per port to 2047 PUs per port).

o Support for ACF/VTAM for VSE/ESA networks

NOTE: These enhancements do not pertain to OSA-1.

LOAD BALANCING, REDUNDANCY, AND OVERFLOW

LOAD BALANCING: A single Medium Access Control (MAC) address can now be defined for attached OSA-2 SNA/APPN Physical Units (PUs), even though connections may be via multiple physical ports. This support is offered for source-route bridged environments only (Token-Ring and FDDI). The number of sessions established through a port is monitored, and user session loads are evenly distributed across the equally configured ports.

REDUNDANCY: A secondary path between the LAN workstation and the host system can now be configured. If the primary path becomes unavailable, the secondary path will receive the LAN traffic. This increases system availability and simplifies network management.

OVERFLOW: User sessions flow through the primary OSA-2 port until the session capacity has been reached. Additional user sessions will automatically flow to the next OSA-2 port. Since all user workstations are identically configured, network administration is simplified and the network becomes more scalable. New users can be added non-disruptively.

Load balancing, redundancy, and overflow support is provided by PTFs for OSA/SF as follows:

o OS/390 and MVS - OW20205/UW34618 03/31/97
o VM/ESA - OW23952/UW37028 03/31/97
o VSE/ESA - Provided with VSE/ESA V2.2.1 04/29/97

INCREASED PHYSICAL UNIT (PU) SUPPORT (VIA OSA/SF):

The architecture has been changed to allow up to a maximum of 2047 PUs per physical port to be defined for OSA-2 Ethernet, Token-Ring and FDDI features instead of the current 255 PUs per port. This enhancement is available for currently installed features, as well as new installations. Actual connectivity may vary based upon user workloads.

Increased Physical Unit (PU) Support is provided by PTFs for OSA/SF as follows:

o OS/390 and MVS - OW23429/UW37210 03/31/97

```
o   VM/ESA          - OW24952/UW37028         03/31/97
o   VSE/ESA         - PQ03091/UQ04224         04/29/97

Increased Physical Unit (PU) Support is provided by PTFs for ACT/VTAM as
follows:

o   ACF/VTAM for OS/390 and MVS

    -   VTAM 4.1 OW14043/UW24904
    -   VTAM 4.2 OW14043/UW24905
    -   VTAM 4.3 OW14043/UW24906

o   ACF/VTAM VM/ESA

    -   VM60877/UV59834

o   ACF/VTAM VSE/ESA

    -   DY44347/UD50254

VSE/ESA - SNA SUPPORT
____________________

OSA-2 and OSA/SF support is delivered via VSE/ESA Version 2 Release 2.1. This
announcement of VSE/ESA support satisfies the Statement of General Direction
contained in Hardware Announcement 196-194, and Hardware Announcement
196-193, dated September 10, 1996.

The OSA-2 feature provides ACF/VTAM for VSE/ESA host applications with direct
access to Ethernet, Token-Ring, and FDDI LANs and Asynchronous Transfer Mode
(ATM) Forum-compliant LAN emulation networks.

OSA/SF is available:

o   As a non-exclusive element of OS/390 Release 1 or above (5645-001)
o   As a separate program product, S/390 Open Systems Adapter Support
    Facility Version 1 Release 2 for MVS/ESA 4.3 or above (5655-104)
o   As a facility of VM/ESA Version 2 Release 2.0 (5654-030)
o   As a component of VSE Central Functions 6.1.1 in VSE/ESA Version 2
    Release 2.1 (5690-VSE).

MORE INFORMATION
________________

Announcements 297-043, 297-040
```

Other Information Sources

This section lists additional sources of information.

Other Publications

For additional information about performance, refer to:

- *DB2 Connect 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 extensive information about DB2 Connect performance tuning, as well as case studies and examples, on the World Wide Web. Set your Web browser to the following URL:

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

Additional Hints and Tips for SNA Users

See "Additional SNA Performance Tuning Hints and Tips" on page 151.

Chapter 13. 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:

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, 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. Go to the following Web page:
   http://www.ibm.com/software/data/db2/library/
2. Select the DB2 Universal Database link.
3. Search for "Technotes" using the keywords "DDCS", or "Connect".

APPC, CPI-C, and SNA Sense Codes Documentation

Documentation on APPC, CPI-C, and SNA Sense Codes that was provided has been repackaged and it is provided as a PostScript file, and an HTML book (English only).

The PDF version of this book can be found on the DB2 publications CD-ROM. The HTML copy of this book is always installed when DB2 Connect is installed, and this book is integrated with the DB2 Information Center.

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 host or AS/400 database 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 host or AS/400 database server and workstation communications configured properly?*
   - There are three configurations that must be considered:
     a. The host or AS/400 database server configuration identifies the application requester to the server. The host or AS/400 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 host or AS/400 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 host or AS/400 server database administrator and the Network administrators have utilities available to diagnose problems.
5. *Do you have the level of authority required by the host or AS/400 server database management system to use the host or AS/400 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 host or AS/400 database server, are you unsuccessful?*
   - Did you follow the procedure to bind the command line processor to the host or AS/400 database server?

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, refer to 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, refer to the *Troubleshooting Guide*. For information about messages found in the log, refer to the *Message Reference*.
- `db2diag.log`

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

  This file is located in `x:\sqllib\db2\db2diag.log` on Intel systems, where `x:` represents the logical drive, and `db2` represents the instance name.
- `db2alert.log` (Same file locations as `db2diag.log`).
- The trace utility, as described in "Trace Utility (ddcstrc)".
- 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 and Windows 2000 systems, use the Event Viewer.

For more information on troubleshooting TCP/IP connections (or other topics), refer to *Troubleshooting Guide*, or search for "Technotes" on the DB2 Product and Service Technical Library (see "Using the World Wide Web" on page 161.

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 host or AS/400 database server 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 host or AS/400 database server, but the command fails and you receive an unsuccessful return code. If you understand exactly what information was conveyed to the host or AS/400 database server 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 host or AS/400 database server management system. Data sent to the host or AS/400 database server is labeled SEND BUFFER and data received from the host or AS/400 database 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 host or AS/400 database server. (For more on mapping, see "Chapter 11. SQLCODE Mapping" on page 121.) 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, refer to:

- *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:

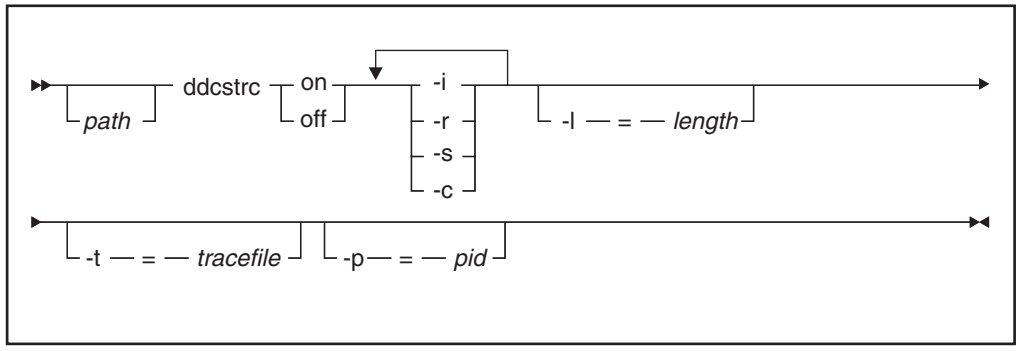


Figure 9. Syntax of the ddcstrc Command

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 DB2 Connect tracing of DRDA flows with the host or AS/400 database server.

off Turns off DB2 Connect tracing of DRDA flows with the host or AS/400 database server.

-i Timestamps will be included in the trace information.

-r Traces DRDA data streams received from the host or AS/400 server system.

-s Traces DRDA data streams sent to the host or AS/400 database server.

-c Traces the SQLCA received from the host or AS/400 database server.

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 "Chapter 8. Database System Monitor" on page 95.

Trace Output

The ddcstrc utility 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.

Notes:

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 host or AS/400 database server. From the user's viewpoint, a `CONNECT TO` database command has been issued using the command line processor.

Figure 10 uses DB2 Connect Enterprise Edition Version 7 and DB2 Universal Database for OS/390 Version 5.1 over an APPC connection.

Figure 11 on page 171 uses DB2 Connect Enterprise Edition Version 7 and DB2 Universal Database for OS/390 Version 5.1 over a TCP/IP connection.

```
1       DB2 fnc_data      gateway_drda_ar      sqljcsend (1.35.10.80)
        pid 95212; tid 537115484; 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  F282974040404040 4040404040404040   ...@@@@@@@@@@@@@  2bp
  0020  4040F0F0F0F1F7F3 C5C3000C116DA685   @@..........m..     000173EC..._we
  0030  81A2859340400013 115AC4C2F240C396   ....@@...Z...@..  asel  ...]DB2 Co
  0040  95958583A340F54B F200141404140300   .....@.K........  nnect 5.2.......
  0050  0414440003240700 05240F0003000D11   ..D..$...$......  ................
  0060  47D8C4C2F261F6F0 F0F00085D0010002   G....a..........  .QDB2/6000.e}...
  0070  007F200100162110 E2C1D56DC6D9C1D5   .. ...!....m....  ."......SAN_FRAN
  0080  C3C9E2C3D6404040 40400006210F2407   .....@@@@@..!.$.  CISCO     ......
  0090  000D002FD8E3C4E2 D8D3C1E2C3000C11   .../............  ....QTDSQLASC...
  00A0  2EE2D8D3F0F5F0F2 F0003C210437E2D8   ..........
```

Figure 10. Example of Trace Output (APPC connection) (Part 1 of 2)

```
3       DB2 fnc_data      gateway_drda_ar      sqljcsend (1.35.10.80)
        pid 95212; tid 537115484; 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 95212; tid 537115484; 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 95212; tid 537115484; node 0; cpid 0; sec 0; nsec 0; tpoint 179
        SQLCA

        SQLCAID:  SQLCA
        SQLCABC:  136
        SQLCODE:  0
        SQLERRML: 0
        SQLERRMC:
        SQLERRP:  DSN
        SQLERRD[0—>5]: 00000000, 00000000, 00000000, 00000000, 00000000, 00000000
        SQLWARN(0—>A):  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,
        SQLSTATE: 00000
```

Figure 10. Example of Trace Output (APPC connection) (Part 2 of 2)

```
1       DB2 fnc_data      gateway_drda_ar      sqljcsend (1.35.10.80)
        pid 80286; tid 537125164; 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  F282974040404040 4040404040404040   ...@@@@@@@@@@@@@  2bp
  0020  4040F0F0F0F1F3F9 F9C5000C116DA685   @@..........m..    0001399E..._we
  0030  81A2859340400013 115AC4C2F240C396   ....@@...Z...@..  asel  ...]DB2 Co
  0040  95958583A340F54B F200181404140300   .....@.K........  nnect 5.2.......
  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  2EE2D8D3F0F5F0F2 F0003C210437E2D8   ..........

3       DB2 fnc_data      gateway_drda_ar      sqljcsend (1.35.10.80)
        pid 80286; tid 537125164; 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 80286; tid 537125164; 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 80286; tid 537125164; node 0; cpid 0; sec 0; nsec 0; tpoint 179
        SQLCA

        SQLCAID:  SQLCA
        SQLCABC:  136
        SQLCODE:  0
        SQLERRML: 0
        SQLERRMC:
        SQLERRP:  DSN
        SQLERRD[0–>5]: 00000000, 00000000, 00000000, 00000000, 00000000, 00000000
        SQLWARN(0–>A):  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,
        SQLSTATE: 00000
```

Figure 11. 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 host or AS/400 database server CCSID(s)
- The host or AS/400 database server 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 host or AS/400 database server 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 host or AS/400 database server 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 host or AS/400 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 has code point X'2104' (see “Implementing Charge-Back Accounting on DB2 Universal Database for OS/390” on page 54).

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 EXCSATRD and ACCRDBRM

CCSID values are also returned from the host or AS/400 database 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 host or AS/400 database server system of 500 (X'01F4', SBCS CCSID).

If DB2 Connect does not recognize the code page coming back from the host or AS/400 database server, SQLCODE -332 will be returned to the user with the source and target code pages. If the host or AS/400 database 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 Universal Database 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 Universal Database 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 host or AS/400 database server management system to commit the current unit of work. The fourth buffer is received from the host or AS/400 database 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 12 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  0065D00300001005F 240800FFFFFF34F4   .e....._$.....4.  ..}....^.......4
  0010  F2F7F0F4C4E2D5E7 D6E3D34000E2C1D5   ..........@....  2704DSNXOTL .SAN
  0020  6DC6D9C1D5C3C9E2 C3D640404040040FF   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
        SQLERRD[0—>5]: FFFFFE0C, 00000000, 00000000, FFFFFFFF, 00000000, 00000000
        SQLWARN(0—>A):  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,
        SQLSTATE: 42704
```

Figure 12. Receive Buffer Example

Most Common DB2 Connect Problems

This section lists the most common symptoms of connection problems encountered when using DB2 Connect. In each case, you are provided with:

- A combination of a message number and a return code (or protocol specific return code) associated with that message. Each message and return code combination has a separate heading, and the headings are ordered by message number, and then by return code.
- A `symptom` is provided, usually in the form of a sample message listing.

- A suggested `solution` is provided, indicating the probable cause of the error. In some cases more than one suggested solution may be provided.

Notes:

1. Refer to your product `Quick Beginnings` manual and the latest Release Notes for the most up-to-date information about recommended software fix levels.
2. For message and return code combinations specific to APPC communications, an SNA sense code may also be indicated. At present, any SNA sense code information associated with a particular message must be obtained from the SNA subsystem.

   Sometimes SNA sense codes can be viewed by looking through system logs. Whether this is the case or not depends on the SNA subsystem being used, and in some situations you may have to recreate the problem with an SNA trace active in order to obtain the sense code information.
3. The term `gateway` refers to DB2 Connect Enterprise Edition.

SQL0965 or SQL0969

Symptom

Messages SQL0965 and SQL0969 can be issued with a number of different return codes from DB2 Universal Database for AS/400, DB2 Universal Database for OS/390, DB2 for MVS/ESA, and DB2 for VM & VSE.

When you encounter either message, you should look up the original SQL code in the documentation for the database server product issuing the message.

Solution

The SQL code received from the host database cannot be translated. Correct the problem, based on the error code, then resubmit the failing command.

SQL1338 During CONNECT

Symptom / Cause

The symbolic destination name was not defined, or it is not properly defined.

For example, this can happen when an APPC node is used and the symbolic destination name specified in the DB2 node directory does not match a CPI-C entry in the local APPC communications subsystem configuration.

Another cause can be that there is more than one SNA stack installed on your machine. You may need to check PATH and LIBPATH to ensure that the stack you want to use is referenced first.

Solutions

1. Make sure the CPIC Side Information profile name specified in the DB2 Node directory entry matches the SNA configuration (it is case sensitive).
2. You may need to check PATH and LIBPATH to ensure that the SNA stack that you want to use is referenced first.

SQL1403N During CONNECT

Symptom

SQL1403N The username and/or password supplied is incorrect.

Solution

1. User fails to authenticate at the DB2 Connect workstation. Determine whether the user is supposed to be authenticated at the DB2 Connect workstation.

   If yes, make sure that the correct password is provided on the CONNECT statement if necessary.

   If no, the system database directory entry must have been incorrectly cataloged using AUTHENTICATION SERVER (this is the default if AUTHENTICATION is not specified explicitly). If this is the case, then recatalog the entry using AUTHENTICATION DCS or CLIENT.
2. Password not available to send to the target server database. If the system database directory entry is cataloged using AUTHENTICATION DCS, then a password has to be flowed from the DB2 Client to the target server database. On certain platforms, for example AIX, the password can only be obtained if it is provided on the CONNECT statement.

SQL5043N

Symptom

Support for one or more communications protocols failed to start successfully. However, core database manager functionality started successfully.

Perhaps the TCP/IP protocol is not started on the DB2 Connect gateway. There may have been a successful client connection previously.

If `diaglevel = 4`, then `db2diag.log` may contain a similar entry, for example:

```
1997-05-30-14.09.55.321092   Instance:svtdbm5   Node:000
PID:10296(db2tcpcm)   Appid:none
common_communication  sqlcctcpconnmgr_child   Probe:46
DIA3205E Socket address "30090" configured in the TCP/IP
services file and
required by the TCP/IP server support is being used by another
process.
```

Solution

This warning is a symptom which signals that DB2 Connect, acting as a gateway for remote clients, is having trouble handling one or more client communication protocols. These protocols can be TCP/IP, APPC and others, and usually the message indicates that one of the communications protocols defined to DB2 Connect is not configured properly.

Often the cause may be that the DB2COMM profile variable is not defined, or is defined incorrectly. Generally, the problem is the result of a mismatch between the DB2COMM variable and names defined in the database manager configuration (for example, svcename, nname, or tpname).

One possible scenario is having a previously successful connection, then getting the SQL5043 error message, while none of the configuration has changed. This could occur using the TCP/IP protocol, when the remote system abnormally terminates the connection for some reason. When this happens, a connection may still appear to exist on the client, and it may become possible to restore the connection without further intervention by issuing the commands shown below.

Most likely, one of the clients connecting to the gateway still has a handle on the TCP/IP port. On each client machine that is connected to the gateway:

1. db2 terminate
2. db2stop

SQL30020

Symptom

SQL30020N Execution failed because of a Distributed Protocol Error that will affect the successful execution of subsequent commands and SQL statements.

Solutions

Service should be contacted with this error.

Check the db2dump directory for an ffdc dump (pid.000). Then, format this dump file with `db2fdump` and look in the result file for "ERROR". An MVS ABEND may be listed here. In this case check the MVS console for further information, and look up the abend code in the DB2 for MVS Messages and Codes manual.

SQL30060

Symptom

SQL30060N "<authorization-ID>" does not have the privilege to perform operation "<operation>".

Solution

When connecting to DB2 for MVS or DB2 for OS/390, the Communications Database (CDB) tables have not been updated properly. Refer to:

- DB2 Connect Quick Beginnings, or
- DB2 Connectivity Supplement

SQL30061

Symptom

Connecting to the wrong host or AS/400 database server location - no target database can be found.

Solution

The wrong server database name may be specified in the DCS directory entry. When this occurs, SQLCODE -30061 is returned to the application.

Check the DB2 node, database, and DCS directory entries. The target database name field in the DCS directory entry must correspond to the name of the database based on the platform. For example, for a DB2 Universal Database for OS/390 database, the name to be used should be the same as that used in the Boot Strap Data Set (BSDS) ″LOCATION=locname″ field, which is also provided in the DSNL004I message (LOCATION=location) when the Distributed Data Facility (DDF) is started. See also “The Database Concept” on page 4, and “Chapter 6. Updating Database Directories” on page 73.

Your DB2 Connect Quick Beginnings manual also contains examples showing how to update the DB2 catalogs. See the ″Update the DB2 Directories″ section in each chapter describing SNA configuration, or see the ″Configuring Host and AS/400 Databases for DB2 Connect″ chapter, and the ″Configuring the TCP/IP Connection″ section.

The correct commands for an APPC or APPN node are:

```
db2 catalog appc node <node_name> remote <sym_dest_name> security program
db2 catalog dcs database <local_name> as <real_db_name>
db2 catalog database <local_name> as <alias> at node <node_name>
            authentication dcs
```

The correct commands for a TCP/IP node are:

```
db2 catalog tcpip node <node_name> remote <host_name_or_address>
            server <port_no_or_service_name>
db2 catalog dcs database <local_name> as <real_db_name>
db2 catalog database <local_name> as <alias> at node <node_name>
            authentication dcs
```

To connect to the database you then issue:

```
db2 connect to <alias> user <user_name> using <password>
```

SQL30073 with Return Code 119C During CONNECT

Symptom

Message SQL30073 is issued with return code 119C. This happens when the target server database does not support the code page used by the DB2 client (going through DB2 Connect). The code page is derived from the configuration of the operating environment in which the DB2 client is running.

See the *Administration Guide* for further information.

Solution

This problem can often be resolved by installing a fix at the target server database system. Contact the appropriate service organization and obtain and apply any fix which might be recommend for this symptom.

As a temporary workaround, the user can override the default code page by setting the DB2CODEPAGE environment variable. Check the locale or set DB2CODEPAGE=850.

On UNIX platforms, the user may be able to switch to a different code page by setting the LANG environment variable to a different value.

SQL30081N with Return Code 1

Symptom

Symptom is the following message plus an SNA sense code:

```
db2 connect to <database name> user <userid>
Enter password for <userid>:
SQL30081N  A communication error has been detected.
Communication protocol
being used: "APPC".  Communication API being used: "CPI-C".
Location where
the error was detected: "".  Communication function detecting
the error:
"cmallc".  Protocol specific error code(s): "1", "*",
"0x10030021".
SQLSTATE=08001
```

Solution(s)

In this sample the sense code is 10030021.

The most common sense codes associated with this error message, and the suggested solution in each case, are as follows:

1.

```
SQL30081N with return code 1 and sna sense code 0877002C
```

Wrong network name has been specified.

2.

```
SQL30081N with return code 1 and SNA sense code ffff0003
```

The wrong MAC address has been specified or the SNA link is not active.

3.

```
SQL30081N with return code 1 and SNA sense code 10030021
```

There is an LU type mismatch.

4.

```
SQL30081N with return code 1 and SNA sense code 084B6031
```

The MAXDBAT in DSNZPARM (at a DB2 for MVS or DB2 for OS/390 host) is set to 0

Other suggestions:

1. When creating the Local LU profile, define the LU as the default LU. For example, in the SNA Feature list panel in CM/2, either:
   - Place a checkmark in the checkbox 'Use this local LU as your default local LU alias', or
   - Set the profile or environment variable `APPCLLU` on the DB2 Connect Enterprise Edition gateway system to the Local LU name. You can do this on OS/2 systems, for example, by editing CONFIG.SYS, or on Windows NT systems via the Control Panel.
2. Check that SNA is started on the DB2 Connect gateway
3. If you are using DB2 for MVS or DB2 for OS/390, check that the Distributed Data Facility (DDF) address space is started and that DB2 is running.

SQL30081N with Return Code 2

Symptom

Message SQL30081N is received with Return Code 2 and SNA Sense Code 08120022.

Solution

The NUMILU parameter at the NCP (host end of the link) may be set to the default (0). Check this. Modify the NCP definition if necessary before retrying, after putting the change into effect.

SQL30081N with Return Code 9

Symptom

Symptom is the following message (the SNA sense code is not required in this case):

```
db2 connect to <database> user <userid>
SQL30081N  A communication error has been detected.
Communication protocol
being used: "APPC".  Communication API being used: "CPI-C".
Location where
the error was detected: "".  Communication function detecting
the error:
"cmsend".  Protocol specific error code(s): "9", "*",
"0x10086021".
SQLSTATE=08001
```

Solution

The problem is that the Transaction Program name (TPNAME) is not defined correctly on the DB2 Connect system. For example, you may have updated your SNA configuration, but not yet verified it at the DB2 Connect gateway. Refer to the *DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings*, or *DB2 Connect Personal Edition Quick Beginnings* manual for further details.

SQL30081N with Return code 10

Symptom

Symptom is the following message (the SNA sense code is not required):

```
SQL30081N  A communication error has been detected.
Communication protocol
being used: "APPC".  Communication API being used: "CPI-C".
Location where
the error was detected: "".  Communication function detecting
the error:
"cmrcv".  Protocol specific error code(s): "10", "*", "*".
SQLSTATE=08001
```

Solution

Check to make sure that DB2 is correctly installed.

If you are using a DB2 Connect for OS/2 gateway, you may see the following if the TP name is not defined properly:

```
Protocol specific error code(s): "10", "*", "0x084C0000".
SQLSTATE=08001
```

For example, in CM/2, in this case it should be defined as follows:

```
Transaction program name        = 'tpname'    (user defined)
OS/2 program path and file name = notused
```

and (on the next CM/2 configuration screen)

```
Presentation type - background
Operation type - Queued, operator preloaded
```

SQL30081N with Return Code 20

Symptom

```
SQL30081N  A communication error has been detected.
Communication protocol
being used: "APPC".  Communication API being used: "CPI-C".
Location where
the error was detected: "".  Communication function detecting
the error:
"xcstp".  Protocol specific error code(s): "20", "*", "*".
SQLSTATE=08001
```

Solution

Ensure that the SNA subsystem is started on the DB2 Connect system.

SQL30081N with Return code 27

Symptom

Message SQL30081N is received with Return Code 27 and SNA Sense Code 800Axxxx.

Solution

The VTAM Path Information Unit (PIU) is too large.

SQL30081N with Return Code 79

Symptom

```
SQL30081N  A communication error has been detected.
Communication protocol
being used: "TCP/IP".  Communication API being used: "SOCKETS".
Location
where the error was detected: "".  Communication function
detecting the error:
"connect".  Protocol specific error code(s): "79", "*", "*".
SQLSTATE=08001
```

Solution(s)

This error can occur in the case of a remote client failing to connect to a DB2 Connect gateway. It can also occur when connecting from the DB2 Connect gateway to a host.

1. The `DB2COMM` profile variable may set incorrectly on the DB2 Connect gateway. Check this. For example, the command `db2set db2comm=tcpip` should appear in `sqllib/db2profile` when running DB2 Extended Enterprise Edition on AIX.
2. There may be a mismatch between the TCP/IP service name and/or port number specifications at the DB2 client and the DB2 Connect gateway. Verify the entries in the TCP/IP `services` files on both machines.
3. Check that DB2 is started on the DB2 Connect gateway. Set the Database Manager Configuration `diaglevel` to 4, using the command:

```
db2 update dbm cfg using diaglevel 4
```

After stopping and restarting DB2, look in the db2diag.log file to check that DB2 TCP/IP communications have been started. You should see output similar to the following:

```
1998-02-03-12.41.04.861119   Instance:svtdbm2   Node:00
PID:86496(db2sysc)   Appid:none
common_communication  sqlcctcp_start_listen   Probe:80
DIA3000I "TCPIP" protocol support was successfully started.
```

SQL30081N with Protocol Specific Error Code 10032

Symptom

```
SQL30081N  A communication error has been detected.
Communication protocol
being used: "TCP/IP".  Communication API being used: "SOCKETS".
Location
where the error was detected: "9.21.85.159".  Communication
function detecting
the error: "send".  Protocol specific error code(s): "10032",
"*", "*".
SQLSTATE=08001
```

Solution

This error message may be received when trying to disconnect from a machine where TCP/IP communications have already failed. Correct the problem with the TCP/IP subsystem.

On most machines, simply restarting the TCP/IP protocol for the machine is the way to correct the problem. Occasionally, recycling the entire machine may be required.

Part 3. Appendixes

Appendix A. Functions Delivered in Previous Releases

Beginning with the most recent version and release, this section provides a summary of the enhancements introduced at each version and release is presented.

DB2 Connect Version 6 Release 1

DB2 Connect Version 6.1 included the following enhancements:

- Using the TCP/IP Communications Protocol
- Two-Phase Commit
- Multi-Row Stored Procedures
- DCE Security
- DCE Cell Directory Support and Host Systems
- Enhanced Security Failure Notification
- Enhanced System/390 SYSPLEX Exploitation
- Optimized Catalog Access for ODBC and JDBC Applications
- New BIND Options
- Microsoft Transaction Server Support
- Simplified Password Management
- Client Information Enhancements
- Bidirectional Language Support
- Monitoring DB2 Connect Applications
- Two-Phase Commit Enhancements
- Simplified DB2 Syncpoint Manager Configuration
- Additional Data Objects and Types Supported
- DB2 Connect for Personal Communications

DB2 Connect Version 5 Release 2

DB2 Connect Version 5.2 included the following enhancements for the host and AS/400 DRDA functions:

- DCE Cell Directory support
- Enhanced security failure notification
- Enhanced System/390 SYSPLEX exploitation
- Optimized Catalog Access for ODBC and JDBC Applications
- Microsoft Transaction Server support

- New BIND options (DYNAMICRULES)
- Set Client Information API enhancements
- SQLDescribeParam support for DB2 Connect
- Support for Bidirectional Languages
- System Monitor enhancements
- Two-phase commit support enhancements
- Simplified DB2 Syncpoint Manager Configuration
- Support for the SCO** operating system
- Support for Big Integer, Large Object, Row ID, and User Defined Distinct data types.

DB2 Connect Version 5.0

- 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 Universal Database for OS/390 Version 5.1 or higher)
  - 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 Universal Database for OS/390 Version 5.1 or higher)
- Usability:
  - New installation method
  - TCP/IP database connections are much easier to configure (requires DB2 Universal Database for OS/390 Version 5.1 or higher, or DB2 Universal Database for AS/400 Version 4.2)
  - 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 Universal Database for OS/390 Version 5.1 or higher)
  - 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 Universal Database for OS/390 Version 5.1 or higher)
- Connectivity:
  - Support for DRDA over TCP/IP connections to other IBM DRDA Application Servers, as they introduce support for TCP/IP.
  - 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 Universal Database for OS/390 Version 5.1 or higher)
  - Ability for desktop applications to participate in a 2-phase commit transactions without the need for a gateway (TCP/IP only, requires DB2 Universal Database for OS/390 V5.1 or higher)
  - Ability to use DB2 Universal Database for OS/390 for added reliability of transaction coordination (requires DB2 Universal Database for OS/390 Version 5.1 or higher, and TCP/IP)

- Numerous other enhancements and fixes affecting all aspects of system performance, reliability, and usability.

DDCS Version 2 Release 4

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

- A Data Source Setup tool to help you define connections to host and AS/400 servers quickly and easily.
- Wall Data Rumba, 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 Universal Database for AS/400 Version 3.1 servers. See “Stored Procedures” on page 50.
- Able to work with multiple databases in a single transaction.
- Able to improve performance by concatenating SQL statements. See “NOT ATOMIC Compound SQL” on page 52 and “Using Import and Export Utilities” on page 108.
- Able to implement chargeback accounting by using accounting strings. See “Implementing Charge-Back Accounting on DB2 Universal Database for OS/390” on page 54.

- Able to use many new bind options when binding applications to a DRDA application server. See "The BIND Command" on page 92.
- When using a DCE directory, the ability to consolidate the directory information needed by all your clients in a central repository. See "Appendix D. Using DCE Directory Services" on page 199.
- Greater flexibility in SQLCODE processing. See "Chapter 11. SQLCODE Mapping" on page 121.
- Diagnostic information stored in a readable format and consolidated in a single location (the first failure service log). For more information, refer to 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.

Appendix B. Directory Customization Worksheet

Use this worksheet to customize your directories. See "Updating the Directories" on page 84 or refer to the *Command Reference* for command syntax.

Table 9. 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 446)	
Security type	PROGRAM for APPC Nodes; NONE for TCP/IP nodes.	

Notes:

1. The default TCP/IP port number for DRDA is 446
2. Unless you know that the host or AS/400 database server supports SECURITY SOCKS, do not specify SECURITY for a TCP/IP node.

Table 10. DCS Directory Parameters

Parameter	Example	Your value
Database name	DB2DB	
Target database name	NEW_YORK3	
Application requester		
Parameter string	",,,,,LOCALDATE=\"\"YYMMDD\"\"\"	

Table 11. System Database Directory Parameters

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

Appendix C. National Language Support Considerations

DB2 Connect has the following national language support (NLS) considerations:

- DB2 Connect messages are translated into certain languages. For information about accessing translated messages, refer to the *Quick Beginnings* book for your platform.
- DB2 Connect supports a large number of languages and code pages. For a list of these code pages, refer to the *Administration Guide*.
- When data is transferred between DB2 Connect and a host or AS/400 database server, it is usually converted from a workstation code page to a host CCSID (and vice versa).

Further information about using DB2 Connect can be found in the *DB2 Connect Quick Beginnings* books, 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 host or AS/400 database 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 Universal Database for OS/390 data, the following happens:

1. DB2 Connect sends an SQL statement and input data to OS/390.
2. DB2 Universal Database for OS/390 converts the data to an EBCDIC CCSID and processes it.
3. DB2 Universal Database 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 12. Workstation Code Page to Host CCSID Conversion

Host CCSIDs	Code Page	Countries
037, 273, 277, 278, 280, 284, 285, 297, 500, 871, 1140-1149	437, 819, 850, 858, 860, 863, 1004, 1051, 1252, 1275	Albania, Australia, Austria, Belgium, Brazil, Canada, Denmark, Finland, France, Germany, Iceland, Ireland, Italy, Latin America, Netherlands, New Zealand, Norway, Portugal, South Africa, Spain, Sweden, Switzerland, UK, USA
423, 875	737, 813, 869, 1253, 1280	Greece
870	852, 912, 1250, 1282	Croatia, Czech Republic, Hungary, Poland, Romania, Serbia/Montenegro (Latin), Slovakia, Slovenia
1025	855, 866, 915, 1251, 1283	Bulgaria, FYR Macedonia, Russia, Serbia/Montenegro (Cyrillic)
1026	857, 920, 1254, 1281	Turkey
424	862, 916, 1255	Israel - see note 3 below
420	864, 1046, 1089, 1256	Arabic countries - see note 3 below
838	874	Thailand
930, 939, 5026, 5035	932, 942, 943, 954, 5039	Japan
937	938, 948, 950, 964	Taiwan

Table 12. Workstation Code Page to Host CCSID Conversion (continued)

Host CCSIDs	Code Page	Countries
933, 1364	949, 970, 1363	Korea
935, 1388	1381, 1383, 1386	People's Republic of China
1112, 1122	921, 922	Estonia, Latvia, Lithuania
1025	915, 1131, 1251, 1283	Belarus
1123	1124, 1125, 1251	Ukraine

Notes:

1. Code page 1004 is supported as code page 1252.
2. 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, and for some newer single-byte code pages, characters that do not exist in both the source and the target may be mapped to substitution characters and then lost when the data is converted back to the original code page.
3. For bidirectional languages, a number of special ″BiDi CCSIDS″ have been defined by IBM and are supported by DB2 Connect Version 7.

   If the bidirectional attributes of the database server are different from those of the client you can use these special CCSIDS to manage the difference.

   Refer to the *Administration Guide* for details of these special CCSIDs. Refer to the Release Notes for DB2 Connect Version 7 for detailed information about how to set them up for DRDA host connections.

Appendix D. 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. CDS is supported for all DB2 Universal Database clients and DB2 Connect Enterprise Edition on all platforms.

Note: If you want to use DCE Cell 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 host or AS/400 database server that you will access, before creating these objects, you should:

- Ensure 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:* represents the drive name.

  On a Windows 32–bit system
  the filename is *x:\root\dcelocal\etc\cds_attributes* where *x:* represents the drive name and *root* represents the directory where you installed DCE.

```
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
```

- Ensure you have logged into DCE with sufficient authority to create the objects. The following DCE command can be used to login on a UNIX or Windows 2000 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 host or AS/400 database server and workstations. This information is described in the appropriate *Quick Beginnings* book.

Creating a Database Object

The *database object* defines the host or AS/400 database server to DB2 Connect; it must always be defined. For each host or AS/400 database server 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 host or AS/400 database 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 206.

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 server and the host or AS/400 database server:

- For the communication protocol APPC:
  1. The communication protocol (APPC)
  2. The network ID of the host or AS/400 database server
  3. The LU name for the host or AS/400 database server
  4. The transaction program name for connections to the host or AS/400 database server. For DB2 for MVS/ESA, specify DB2DRDA. For any other operating system, specify a valid value that is not in hexadecimal format.
  5. The mode name
  6. The security type, as described in "Security with DCE Directory Services" on page 206. For example:

     ```
     APPC;SPIFNET;NYM2DB2;DB2DRDA;IBMRDB;PROGRAM
     ```

- For the communication protocol TCPIP:
  1. The communication protocol (TCPIP)
  2. The destination TCP/IP hostname (for the host or AS/400 database server).
  3. The TCP/IP port number.
  4. 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
```

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
```

Then, enter the command:

```
cdscp < filename
```

Creating a Database Locator Object

The *database locator object* is used to define a DB2 Connect Enterprise Edition server 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
Following is the information that you need to set up each communication protocol between the DB2 Connect workstation and remote clients.

The following summarizes protocol support by platform:

- On OS/2: APPC, IPX, NETBIOS, and TCP/IP
- On Windows 32-bit operating systems: APPC, IPX, NETBIOS, NPIPE, and TCP/IP
- On UNIX: APPC and TCP/IP

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 206.

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 Named Pipes:

1. The communication protocol (`NPIPE`)
2. The `computer name` of the DB2 Connect workstation.
3. The instance name.

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
```

Then, enter the command:

```
cdscp < filename
```

On Windows 32-bit operating systems, you can specify a named pipe in a similar manner. For example:

```
add    object /.../cdscell1/subsys/database/DBAIX01 DB_Communication_Protocol=\
           NPIPE;computer_name;instance_name
```

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

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

Creating a Routing Information Object

The *routing information 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 host or AS/400 database 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 206.

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 75.

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
```

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* represents the default path used to form the complete name of target databases (default `/.:/subsys/database/`)
- *loc_obj* identifies the client in the DCE name space
- DIR_TYPE DCE specifies that DCE directories are used by the client application
- *route_obj* represents the name of the routing information object (e.g. ROUTE1).
- *protocol* represents 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; for Windows 32-bit operating systems, APPC, TCPIP, IPXSPX, NETBIOS, and NPIPE).
- 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 13 and Table 14 on page 207 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 13. Valid Security Scenarios with DCE using APPC connections

	Database object of the Server		**Routing object**	**Validation**
Case	**Authentication**	**Security**	**Authentication at DB2 Connect Gateway (1=true, 0=false)**	
1	CLIENT	SAME	0	Remote client (or DB2 Connect workstation)
2	CLIENT	SAME	1	DB2 Connect workstation

Table 13. Valid Security Scenarios with DCE using APPC connections (continued)

	Database object of the Server		Routing object	Validation
Case	**Authentication**	**Security**	**Authentication at DB2 Connect Gateway (1=true, 0=false)**	
3	SERVER	PROGRAM	0	host or AS/400 database server
4	SERVER	PROGRAM	1	DB2 Connect workstation and host or AS/400 database server
5	DCE	NONE	N/A	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 14. 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	host or AS/400 database server
4	N/A	N/A	None
5	DCE	N/A	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 users are expected to be authenticated at the location they first sign 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 server but not to the host or AS/400 database server.
- In the third case, the user name and password are validated at the host or AS/400 database server only. The password is sent across the network from the remote client to the DB2 Connect server and from the DB2 Connect workstation to the host or AS/400 database server.
- In the fourth case, the user name and password are validated at both the DB2 Connect workstation and the host or AS/400 database server. The password is sent across the network from the remote client to the DB2 Connect server and from the DB2 Connect server to the host or AS/400 database server.

  Because validation is performed in two places, the same set of user names and passwords must be maintained at both the DB2 Connect server and the host or AS/400 database 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 server must belong to the AIX **system** group.
3. Access to a host or AS/400 database 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 E. 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 host or AS/400 database server:

- If the old client was used with a previous release of DB2 Connect against the same host or AS/400 database 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 host or AS/400 database 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 server, bind each bind list file to each host or AS/400 database 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 host or AS/400 database server database, *bindfile* represents the name of the bind list file, and *path* represents 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 F. Tuning CLI/ODBC Application Performance with the CLISCHEMA Keyword

This section contains new information to help you tune the performance of your ODBC/CLI applications using the CLISCHEMA initialization keyword. It does not contain general information about tuning network or database performance (see "Chapter 12. Performance" on page 127). Here is a summary of the information that follows:

Target Environment

The information presented here is intended primarily for users of DB2 Universal Database for OS/390, and the target environment comprises:

- A CLI/ODBC application running with a DB2 Universal Database client
- DB2 Connect Version 5 or higher (Personal Edition or Enterprise Edition)
- DB2 Universal Database for OS/390 Version 5.1 or higher (or DB2 for MVS/ESA except where indicated otherwise).

CLI/ODBC

CLI/ODBC is an SQL application programming interface that can be called by your database applications. It passes dynamic SQL statements as database function calls. Unlike embedded SQL it does not require host variables or a precompiler.

When an application program calls CLI/ODBC, the first thing that it must do is to make SQL calls to some of the system catalog tables on the target database in order to obtain information about other database contents. CLI/ODBC applications always access the system catalog tables in this way. There are ten API calls that may be made in order to gather information about the database that is being connected to. These API calls include:

- SQLTables
- SQLColumns
- SQLSpecialcolumns
- SQLStatistics
- SQLPrimarykeys
- SQLForeignkeys
- SQLTablePrivileges
- SQLColumnPrivileges
- SQLProcedures
- SQLProcedureColumns.

Further information about these API calls and the tables, refer to the *CLI Guide and Reference*.

By default, when you connect to a database, your CLI/ODBC application will query the system catalog tables for information about *all* the database tables in that database. Especially on a large system this can result in a lot of network traffic and considerable delays when starting an application.

The DB2 CLISCHEMA Initalization Keyword

DB2 Universal Database provides several CLI/ODBC initialization keywords that can be used to limit the amount of data that will be returned by the initial API calls during the ″information gathering″ stage after the database is first connected to. These keywords can be set by:

1. Manually editing the db2cli.ini file.
2. Changing ODBC/CLI settings for the database using the Client Configuration Assistant (on those platforms which support it).
3. Updating the database CLI configuration using the DBA Command Line Interface.

The keywords are:

- DBNAME
- TABLETYPE
- SCHEMALIST
- SYSSCHEMA
- CLISCHEMA

With the exception of the information for CLISCHEMA, these keywords are documented in the CLI/ODBC help and *CLI Guide and Reference*. The rest of this discussion relates only to the use of CLISCHEMA.

Here is the documentation for CLISCHEMA that will be added to *CLI Guide and Reference* in due course:

```
db2cli.ini Keyword Syntax:  CLISCHEMA = clischema
```

```
Default Setting:            No alternatives specified.

DB2 CLI/ODBC Settings Tab:  Not present.
```

Usage Notes

The CLISCHEMA option indicates an alternative schema, tables, and index set to be searched instead of the SYSIBM (or SYSTEM, QSYS2) schemas when the DB2 CLI and ODBC Catalog Function calls are issued to obtain catalog information.

For example, if you specify CLISCHEMA='SERGE', the internal CLI/ODBC API calls that normally reference the system tables will reference the following user tables instead:

- SERGE.TABLES
- SERGE.COLUMNS
- SERGE.SPECIALCOLUMNS
- SERGE.TSTATISTICS
- SERGE.PRIMARYKEYS
- SERGE.FOREIGNKEYS
- SERGE.TABLEPRIVILEGES
- SERGE.COLUMNTABLES
- SERGE.PROCEDURES
- SERGE.PROCEDURESCOLUMNS.

These user tables must be built by the database administrator before CLISCHEMA can be used.

Note: DataPropagator provides support for CLISCHEMA, so that the Database Administrator can perform this task in three possible ways:

1. Using db2cli.exe on the client.
2. Automatically on the server using DataPropagator.
3. Manually on the server.

The information which follows explains how this task can be performed on the client.

db2cli and bldschem Utilities

A utility to set up the user tables required by CLISCHEMA is provided in the form of the previously undocumented bldschem support command of the CLI Command Line Interface, which can be found as: /samples/cli/db2cli.exe. Documentation for db2cli.exe, not including the bldschem support command, can found in: /samples/cli/INTCLI.DOC.

For example, to build the set of user tables that is required to work with CLISCHEMA='SERGE' for the table name STAFF owned by the schema owner (creator) USERID, in the database SAMPLE, you would run the following command after issuing db2start and after registering the database to ODBC/CLI:

```
db2cli < addstaff.txt
```

Where "addstaff.txt" contains the following script:

```
opt callerror on
opt echo on
quickc 1 1 sample userid password
#
# Repeat next line for each table to add.
#
bldschem 1 SERGE USERID STAFF
#
# Exit
#
killenv 1
```

This script will result in the creation of the set of tables SERGE.* as listed above, with indexes, populated using the system catalog table data for the table USERID.STAFF. For example, SERGE.TABLES will be populated with a new row for each entry that is matched. Additional bldschem calls result in appends to the existing SERGE.* tables, with replacement of existing rows.

In summary, the syntax of the bldschem support command is:

```
bldschem <handle_number> <value_of_CLISCHEMA> <schema_owner> <table_name>
```

Where:

```
- <handle_number> should be 1

- <value_of_CLISCHEMA> should be the same as the schema name specified with
     the CLISCHEMA keyword

- <schema_owner> is the creator of the table

- <table_name> can be the name of a user table, a view, an alias, a synonym,
     or a system table name.
   (Wildcard characters are allowed).
```

If you subsequently run the following example through db2cli.exe, then you will append to the user tables SERGE.* created in the previous example, adding rows which reflect the data in the system catalog tables for each table for which FRED and BERT are the schema owners.

```
bldschem 1 SERGE FRED %
bldschem 1 SERGE BERT %
```

When the CLISCHEMA CLI/ODBC keyword is subsequently set to SERGE, processing by ODBC/CLI applications against the SAMPLE database will reference the SERGE.* set of tables instead of the system catalog tables.

Suggested Approach

In most production environments the default search of the system catalog tables can return a very large amount of data, so that each time a CLI/ODBC application opens a database there can be a considerable delay. Even on a typical test database, the delay can easily be of the order of 25 seconds or so.

Measure this delay initially without having any of the above CLI keywords set, while remembering to discount connect time and especially the long delays that can occur when a DB2 client issues its first ever connection to a new database - autobinding can often take several minutes.

How you proceed next depends on the structure of your data and your organization. In some cases you can use DBNAME, SCHEMALIST, and TABLETYPE in combination to limit the search for use by a particular application or group of applications. For example, if production DBA clients usually access tables under a given DBNAME and schema then this is easy to specify.

CLISCHEMA provides the best performance advantages for most users. For this reason we would generally recommend using CLISCHEMA in a production environment, since it is much easier to set up and modify CLISCHEMA's user tables through the CLI command Line Interface (db2cli.exe) and its bldschem support command. See also "db2ocat Catalog Optimizer Tool" on page 216.

Additional Hints and Tips

The CLISCHEMA keyword has to be added to the db2cli.ini file within a section for the DSN name or the common section. A section is text in square brackets. The COMMON section is indicated by the text ″COMMON″ in square brackets. Note that keywords and section names are not case-sensitive.

At connect, each possible keyword is checked for first under the DSN name, then if not found, under the COMMON section. This allows for both DSN specific keywords and global (client) keywords.

Also, the DBALIAS keyword can be used to create different DSN (ODBC Data Sources) that map to the same database. (A DSN name can be up to 255 characters in length, and it is mapped to the 8 char dbname).

In the example below, any time a user connects to TESTDB or any DSN that is not listed in the file, they will use clischema=ODBCCAT. If they connect to TestDBcar2, they will use clischema=odbccat2, but still connect to the testdb database.

Example db2cli.ini file:

```
[TESTDB]

[COMMON]
clischema=odbccat

[TestDBcat1]
DBALIAS=testdb
clischema=odbccat1

[TestDBcat2]
DBALIAS=testdb
clischema=odbccat2
```

db2ocat Catalog Optimizer Tool

A new tool `db2ocat` is provided on Windows 32-bit operating systems to help you optimize system catalog searches for ODBC and JDBC applications.

You can obtain the `db2ocat` point-and-click catalog optimizer utility by downloading `db2ocat.zip` from:

`ftp://ftp.software.ibm.com/ps/products/db2/tools.`

Additional Information Sources

The following additional information sources may be of interest:

- Paper on automatic catalog propagation using DataPropagator:

  `http://www.ibm.com/software/data/db2/os390/odbcatlg.html`

- Paper on manual approach not using DataPropagator:

  `http://www.ibm.com/software/data/db2/os390/odbcmanu.html`

Appendix G. Additional and Related Information Sources

Other Related Publications

Form number	Book title
SG24-2006	*Migrating to DB2 Universal Database Version 5*
SG24-2213	*DB2 for OS/390 Version 5 Performance Topics*
SG24-4893	*DB2 Meets NT*
SG24-4894	*The Universal Connectivity Guide to DB2*
SG24-4693	*Getting Started with DB2 Stored Procedures*
SG24-2212	*DRDA Support for TCP/IP in DB2 Universal Database 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*
GC09-2952	*DB2 Connect Enterprise Edition for UNIX Quick Beginnings*
GC09-2953	*DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings*
GC09-2967	*DB2 Connect Personal Edition Quick Beginnings*
GG24-4155	*Distributed Relational Database Architecture: Using DDCS for AIX DRDA support with DB2 for MVS/ESA and DB2 Universal Database for AS/400*
GG24-4311	*Distributed Relational Database Architecture Cross Platform Connectivity and Application*
SC23-2443	*Encina for AIX Product Family Overview*

Appendix H. Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make

improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
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.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information 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.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Trademarks

The following terms, which may be denoted by an asterisk(*), are trademarks of International Business Machines Corporation in the United States, other countries, or both.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

The following terms are trademarks or registered trademarks of other companies:

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

Java or all Java-based trademarks and logos, and Solaris are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, other countries, or both.

UNIX is a registered trademark in the United States, other countries or both and is 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

Special Characters

Numerics

A

B

C

D

E

F

G

Q

R

S

T

U

V

W

X

Z

Contacting IBM

If you have a technical problem, please review and carry out the actions suggested by the *Troubleshooting Guide* before contacting DB2 Customer Support. This guide suggests information that you can gather to help DB2 Customer Support 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.

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

- 1-800-237-5511 for customer support
- 1-888-426-4343 to learn about available service options

Product Information

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

- 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.

http://www.ibm.com/software/data/
The DB2 World Wide Web pages provide current DB2 information about news, product descriptions, education schedules, and more.

http://www.ibm.com/software/data/db2/library/
The DB2 Product and Service Technical Library provides access to frequently asked questions, fixes, books, and up-to-date DB2 technical information.

Note: This information may be in English only.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
The International Publications ordering Web site provides information on how to order books.

http://www.ibm.com/education/certify/
The Professional Certification Program from the IBM Web site provides certification test information for a variety of IBM products, including DB2.

ftp.software.ibm.com
Log on as anonymous. In the directory `/ps/products/db2`, you can find demos, fixes, information, and tools relating to DB2 and many other products.

comp.databases.ibm-db2, bit.listserv.db2-l
These Internet newsgroups are available for users to discuss their experiences with DB2 products.

On Compuserve: GO IBMDB2
Enter this command to access the IBM DB2 Family forums. All DB2 products are supported through these forums.

For information on how to contact IBM outside of the United States, refer to Appendix A of the *IBM Software Support Handbook*. To access this document, go to the following Web page: http://www.ibm.com/support/, and then select the IBM Software Support Handbook link near the bottom of the page.

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

Printed in the U.S.A.

SC09-2954-00