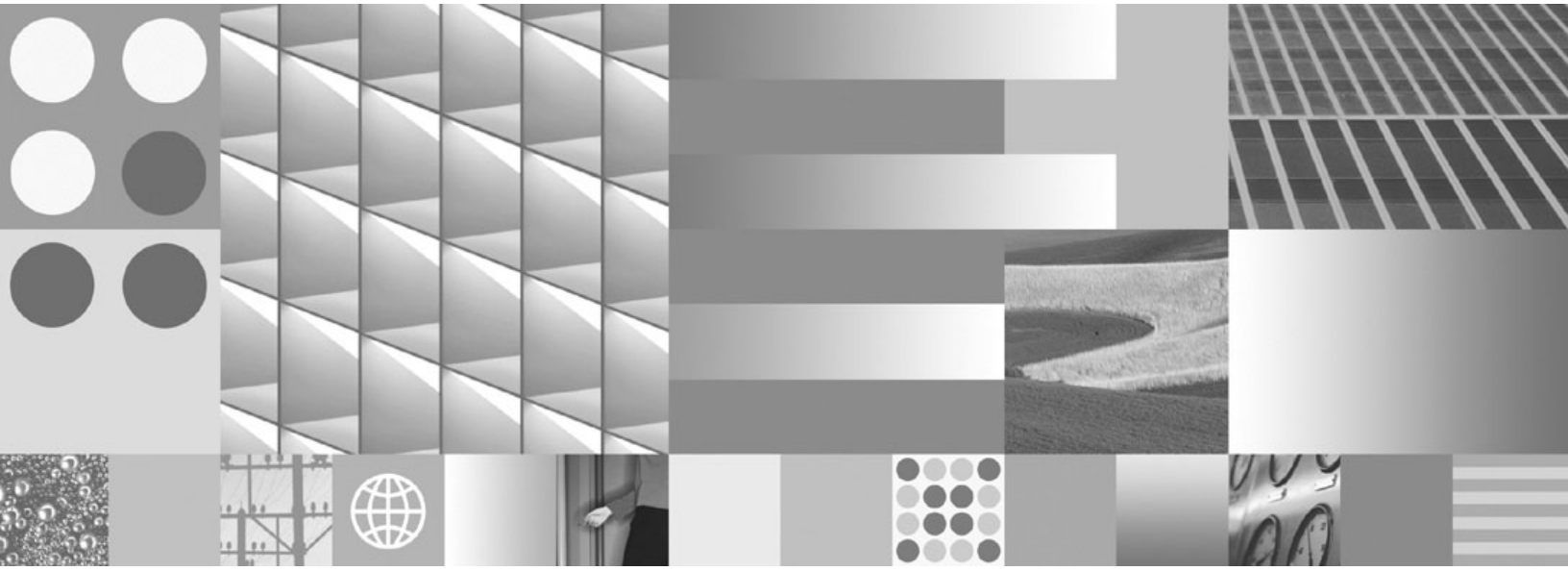
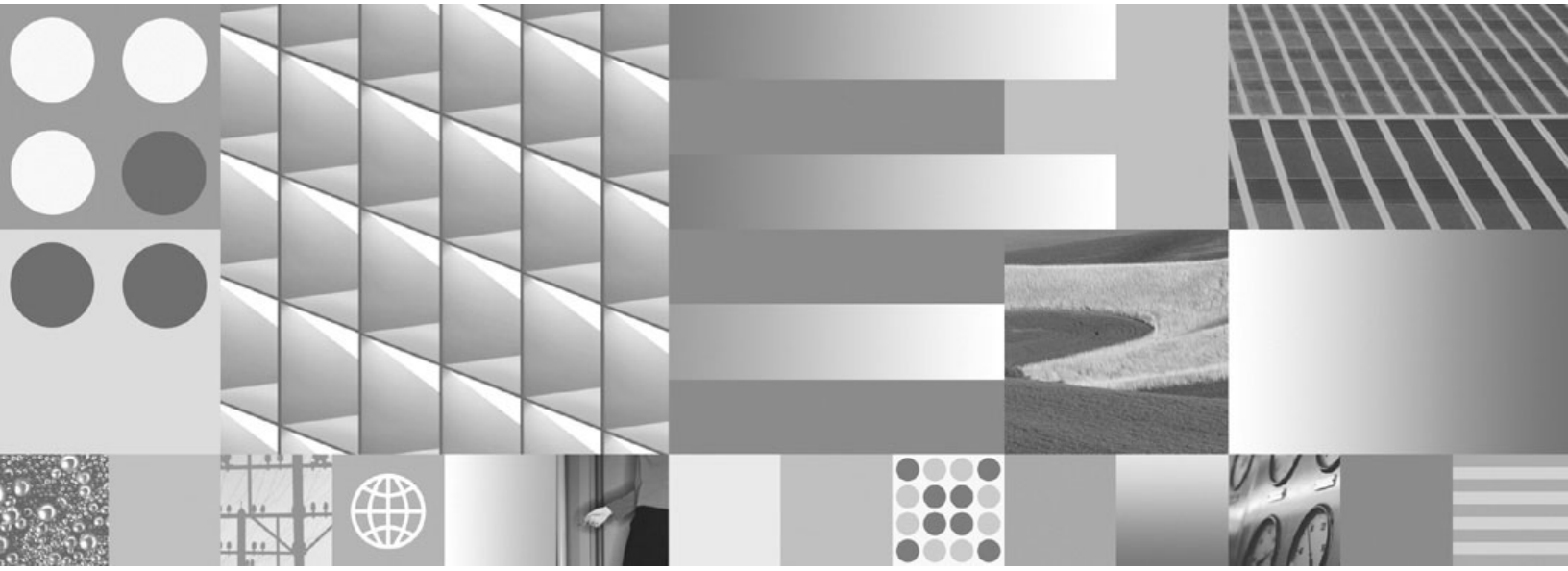
DB2 Connect Version 9.5



DB2 Connect User's Guide
Updated March, 2008

SC23-5841-01

DB2 Connect Version 9.5



DB2 Connect User's Guide
Updated March, 2008

SC23-5841-01

Note

Before using this information and the product it supports, read the general information under Appendix B, "Notices," on page 163.

Edition Notice

This document contains proprietary information of IBM. It is provided under a license agreement and is protected by copyright law. The information contained in this publication does not include any product warranties, and any statements provided in this manual should not be interpreted as such.

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

About this book

The *DB2 Connect User's Guide* provides all the information you need to learn about and use the DB2 Connect™ product. DB2 Connect concepts are presented with typical scenario showing the relationships between DB2 Connect and the other parts of the network environment. Considerations involving database directories, security between systems, multisite updates, moving data, and monitoring DB2 Connect are discussed. How DB2 Connect supports high availability in your network environment is presented. Ensuring good performance by DB2 Connect and across the network is introduced as are some topics concerned with troubleshooting possible problems.

Who should use this book?

System administrators, database administrators, and system communications specialists would all be interested in part or all of this book.

Part 1. DB2 Connect concepts

Chapter 1. DB2 Connect

DB2 Connect provides fast and robust connectivity to host and System i™ databases for e-business and other applications running under Linux™, UNIX®, and Windows® operating systems.

DB2 Connect Personal Edition provides direct connectivity to host and System i DB2® servers, while DB2 Connect server products provide indirect connectivity that allows clients to access host and System i DB2 servers through the DB2 Connect gateway. A variety of DB2 Connect server products provides unique packaging and licensing solutions which allows you to select a product that is appropriate for your environment.

DB2 Connect product offerings

DB2 Connect has several connection solutions, including DB2 Connect Personal Edition, and a number of DB2 Connect server products:

- DB2 Connect Enterprise Edition
- DB2 Connect Application Server Edition
- DB2 Connect Unlimited Edition for zSeries™
- DB2 Connect Unlimited Edition for i5/OS

For detailed information about DB2 Connect product offerings, see http://www.ibm.com/support/docview.wss?rs=73&uid=swg21219983

Functions delivered in Version 9 and previous releases

This section provides a summary of the enhancements introduced at each version and release is presented.

Functions delivered in DB2 Connect Version 9

DB2 Connect Version 9 includes the following enhancements:

- Client support for trusted connections

  A client can create trusted connections using ODBC, XA, or new Java™ methods to database servers (currently only DB2 for z/OS®) that support trusted contexts. The user name of the client can then be switched without the database server having to fully authenticate the new name.
- BINARY, VARBINARY, and DECFLOAT data type support

  DB2 for z/OS now supports data types BINARY, VARBINARY, and DECFLOAT. Support for these data types has been added to DB2 CLI and IBM Data Server Provider for .NET. Your applications using DB2 Connect to access DB2 for z/OS can use DB2 CLI and IBM Data Server Provider for .NET to take advantage of the new data types. A new connection setting named SQL_ATTR_DECFLOAT_ROUNDING_MODE allows the client to specify what type of rounding should occur if any operations on the server side require rounding of a decimal float value.
- NetBIOS and SNA communications protocols are no longer supported

  Customers using these protocols need to re-catalog their nodes and their databases using a supported protocol such as TCP/IP.

- IPv6 communication protocol support has been added

  Support has been added for Internet Protocol Version 6 (IPv6) so that you can now connect to servers using either IPv4 or IPv6 addresses.
- Command Line Processor (CLP) 64 KB limit for SQL statements is removed

  A new command line processor (CLP) limit of approximately 2 MB for SQL statements and for CLP commands containing SQL statement components is comparable with the limits on the other DB2 tools. Your applications using DB2 Connect can now take advantage of this new limit.
- IBM Data Server Provider for .NET enhancements including .NET Framework 2.0 support

  This support and enhancements will help you to develop more powerful .NET applications for use with DB2 Connect. Some of the new capability includes:
  - Applications can fetch a specific set of rows instead of having to scroll through an entire result set.
  - Applications can perform a bulk data copy operation.
  - Applications can determine the number of SQL statements to collect before using them as a batch to the DB2 database server. This will result in fewer individual transmissions of data between the client application and the database server.
- Two phase commit for multi-vendor data sources when using WebSphere® Federation Server

  DB2 Connect applications can use the WebSphere Federation Server to reach data sources offered by many IBM® and non-IBM vendors.
- Connection timeout support for database applications

  You are able to limit the amount of time your DB2 Connect database applications should wait for a connection. This is particularly helpful when the target database server is inaccessible.
- DB2 Connect Personal Edition easier to upgrade

  You can upgrade DB2 Connect Personal Edition on the Windows and Linux operating systems by supplying the appropriate Electronic Certificate File. It is no longer necessary to perform an entire installation when upgrading.
- DB2 licensing support changes

  DB2 Connect product packaging changes are part of the enhancements to the License Center and the Licensed Management Tool (db2licm) command.

Functions delivered in DB2 Connect Version 8 Release 2

DB2 Connect Version 8.2 included the following enhancements:

- Automatic Client Reroute

  If a TCP/IP connection to a server or DB2 Connect Server is lost, the client will automatically attempt to reestablish the connection if an alternate server exists. The alternate server is specified on the server instance and its location is sent to the client during the connection.
- Data encryption

  Client/server communication now provides encryption of user data as it travels over the network.

Functions delivered in DB2 Connect Version 8 Release 1 (including all FixPaks and modification levels)

DB2 Connect Version 8.1 included the following enhancements:

- Support for longer SQL statements (up to 2MB)

  SQL statements up to 2 MBs can flow through CLI and JDBC applications. However, the embedded interface remains at the 64K limit.
- Diagnostic information that identifies the origin of an SQL statement

  Provides the ability to determine which application program issued a particular statement into the DB2 for z/OS dynamic SQL statement cache.
- Column-wise input array

  Allows applications to provide multiple sets of parameters for a single SQL statement.
- Monitoring network time

  New monitor elements are used to get a better idea of the database activity and network traffic at the database or application level.
- DB2 CLI dynamic scrollable cursor support

  Dynamic scrollable cursors are now supported in DB2 CLI when accessing servers which are DB2 Universal Database (UDB) for z/OS Version 8.1 or later.
- eWLM support

  Provides the ability to monitor end to end work units through middleware groups to determine bottlenecks.
- Enhancements to the DB2 ping command

  The DB2 ping command now supports the specification of a request and response packet size.

  Note: DB2 Connect does not support the PING command when issued from a Version 7 client through a Version 9 gateway to the host.

Functions delivered in DB2 Connect Version 7 Release 2

DB2 Connect Version 7.2 included the following enhancements:

- Improved support for Microsoft® Transaction Server (MTS) and COM+ technologies
- DB2 Connect Web Starter Kit
- DB2 Connect for Linux on S/390®

Functions delivered in DB2 Connect Version 7 Release 1

DB2 Connect Version 7.1 included the following enhancements:

- XA Concentrator
- Multisite update improvements

Host databases

The term *database* is used throughout this document to describe a relational database management system (RDBMS). Other systems with which DB2 Connect communicates might use the term database to describe a slightly different concept. The DB2 Connect term database can also refer to:

OS/390® or z/OS

DB2 Universal Database (UDB) for OS/390 and z/OS Version 7 or DB2 UDB for z/OS Version 8. A DB2 Universal Database™ for z/OS and

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. LOCATION NAME supports up to 8 alias location names, allowing applications the ability to use different dbalias names to access a Version 8 z/OS server. Use the z/OS -display ddf command to get the DB2 server location name, domain name, IP address and port.

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

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

OS/400®
DB2 for i5/OS, an integral part of the OS/400 operating system. Only one database can exist on an System i server unless the system is configured to use independent auxiliary storage pools.

DB2 Connect and SQL statements

DB2 Connect forwards SQL statements submitted by application programs to host or System i database servers.

DB2 Connect can forward almost any valid SQL statement, as well as the supported DB2 APIs (application programming interfaces):

- JDBC
- SQLJ
- ADO.NET
- OLE DB
- ODBC
- Perl
- PHP
- DB2 CLI
- Embedded SQL

Embedded SQL support

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 System i 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 types are supported by DB2 Connect.

Different host or System i database servers implement SQL differently. DB2 Connect fully supports the common IBM SQL, as well as the DB2 for OS/390 and z/OS, DB2 Server for VSE & VM (formerly SQL/DS™), and DB2 for System i implementations of SQL. IBM SQL is strongly recommended for maintaining database independence.

DB2 Connect administration utilities

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

- The Command Line Processor (CLP) lets you issue SQL statements against a host or System i 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 (CLP).
- Import and export utilities let you load, import, and export data to and from a file on a workstation and a host or System i database server database. These files can then be used for importing data into databases, spreadsheets, and other applications running on your workstation.
- If you are running a DB2 Connect server product, you 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 or z/OS database objects, such as tables, views, buffer pools, and threads.
- The database system monitor utility lets the system administrator monitor system connections. This function is only available when DB2 Connect is acting as server. 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 System i database server.

Note: In previous releases, the DB2 Graphical Administration Tools, such as the Control Center, were supported on all platforms. As of Version 9, the DB2 Graphical Administration Tools are supported only on Windows x86, Windows x64 (AMD64/EM64T), Linux on x86, and Linux on AMD64/EM64T. For all platforms, you can use the DB2 Command Line processor (CLP) for administration purposes.

WebSphere Federation Server and DB2 Connect

WebSphere Federation Server is a separate product offering that provides access to and integration of data across multivendor data sources, while DB2 Connect enables you to leverage the large volumes of data located in existing host and midrange servers.

WebSphere Federation Server helps integrate information by allowing a collection of data sources to be viewed and manipulated as if they were a single source. It makes data source access completely transparent to the calling application. WebSphere Federation Server works in conjunction with DB2 Connect server products. WebSphere Federation Server provides native read and write access to the DB2 family of products, Informix®, Oracle, Sybase, Teradata, and Microsoft SQL Server databases. WebSphere Federation Server also provides read access to nonrelational and life sciences data sources such as BLAST, Documentum, Entrez, IBM Lotus® Extended Search, table-structured files, and XML. You can use it to formulate queries on data in a federated system.

Chapter 2. Distributed Relational Database Architecture

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.

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.

Distributed unit of work

A *distributed unit of work (DUOW)*, also known as multisite update, involves more than one database server within a unit of work. A DUOW has 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 might be multiple requests per unit of work.
- There is one database management server per request.
- Commitment is coordinated across multiple database servers.

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 the Windows, UNIX, and Linux operating systems. These tools are part of the DB2 client. The DB2 database manager supports several programming interfaces: ADO.NET, JDBC, SQLJ, PHP, Perl DBI, embedded SQL, DB2 Call Level Interface (DB2 Call Level Interface), and OLE DB. These APIs can be used by programmers to build applications in a variety of programming languages.

DB2 Connect and DRDA

DB2 Connect implements the DRDA architecture to reduce the cost and complexity of accessing data stored in DB2 Universal Database (UDB) for iSeries, DB2 for System i, DB2 UDB for OS/390 and z/OS, DB2 for z/OS, DB2 Server 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. The AR is the application that is

requesting data. DB2 Connect acts as an application requester on behalf of application programs which can be local to the DB2 Connect workstation or on a separate client remote to DB2 Connect.

An *application server (AS)* is the code that handles the database end of the connection.

DRDA also supports multi-tier connections between an application requester and a server. In this topology, the server that an application requester connects to is an application server, but any other server further downstream is called a database server (DS) as it does not interact directly with the application requester. In addition, to highlight its role as neither the system where a database request originates nor the system that performs the database function for the request, each application server or database server between an application requester and the final database server is also called an intermediate server. The use of database servers and intermediate servers is supported by DB2 Connect.

Figure 1 shows the flow of data between the DB2 Connect workstation and the host or System i server in the case where there are local clients only.

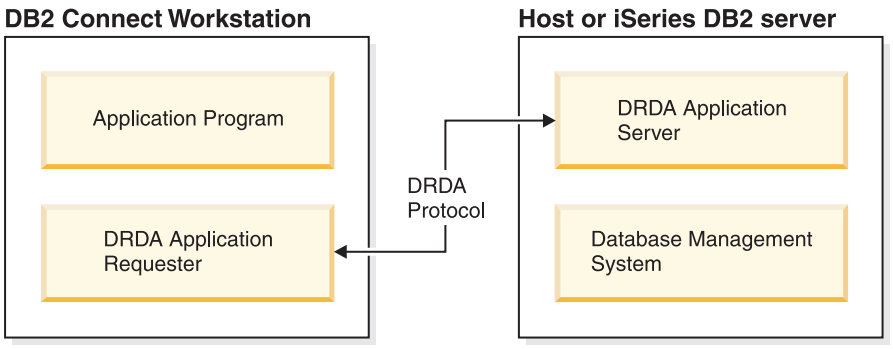


Figure 1. Data flow between a DB2 Connect server and a host or System i server

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

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

These architectures are used as building blocks. The data streams which flow over the network are specified by the 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 might 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 transaction 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).

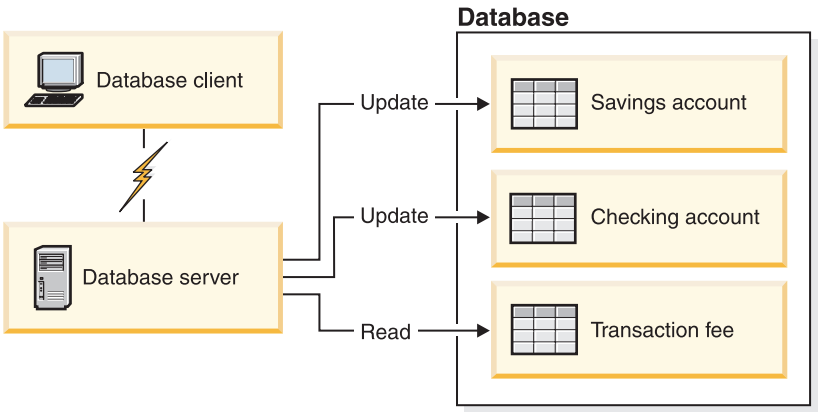


Figure 2. Using a Single Database in a Transaction

To set up such an application, you must:

1. Create the tables for the savings account, checking account and transaction fee schedule in the same database.
2. If physically remote, set up the database server to use the appropriate communications protocol.
3. If physically remote, catalog the node and the database to identify the database on the database server.
4. Precompile your application program to specify a type 1 connection; that is, specify CONNECT(1) on the PREP command.

Distributed requests

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 or z/OS subsystems.

DB2 Connect provides support for distributed requests across databases and DBMSs. For example, you can perform a UNION operation between a DB2 table and an Oracle view. Supported DBMSs include members of the DB2 Family (such as DB2 Database for Linux, UNIX, and Windows, DB2 for OS/390 and z/OS, and DB2 for System i) and Oracle. Multivendor support is available when using DB2 Connect in conjunction with WebSphere Federation Server.

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 consists of a DB2 Connect 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.

Chapter 3. DB2 Connect scenarios

DB2 Connect can provide a variety of solutions to your host or System i database access needs. This topic outlines several scenarios that might apply to your particular needs or environment.

Direct access to host databases

DB2 Connect's basic feature is providing a direct connection to a host database from desktop applications running on Windows, Solaris or Linux workstations. DB2 Connect Personal Edition is the simplest way to provide this solution.

Each workstation that has DB2 Connect Personal Edition installed can establish a direct TCP/IP connection to DB2 Universal Database (UDB) for OS/390 and z/OS, DB2 for z/OS, DB2 UDB for iSeries, DB2 for i5/OS®, and DB2 Database for Linux, UNIX, and Windows 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 two-phase commit protocol.

Figure 3 on page 14 shows a direct connection to a host or System i™ database server from a workstation with DB2 Connect Personal Edition installed.

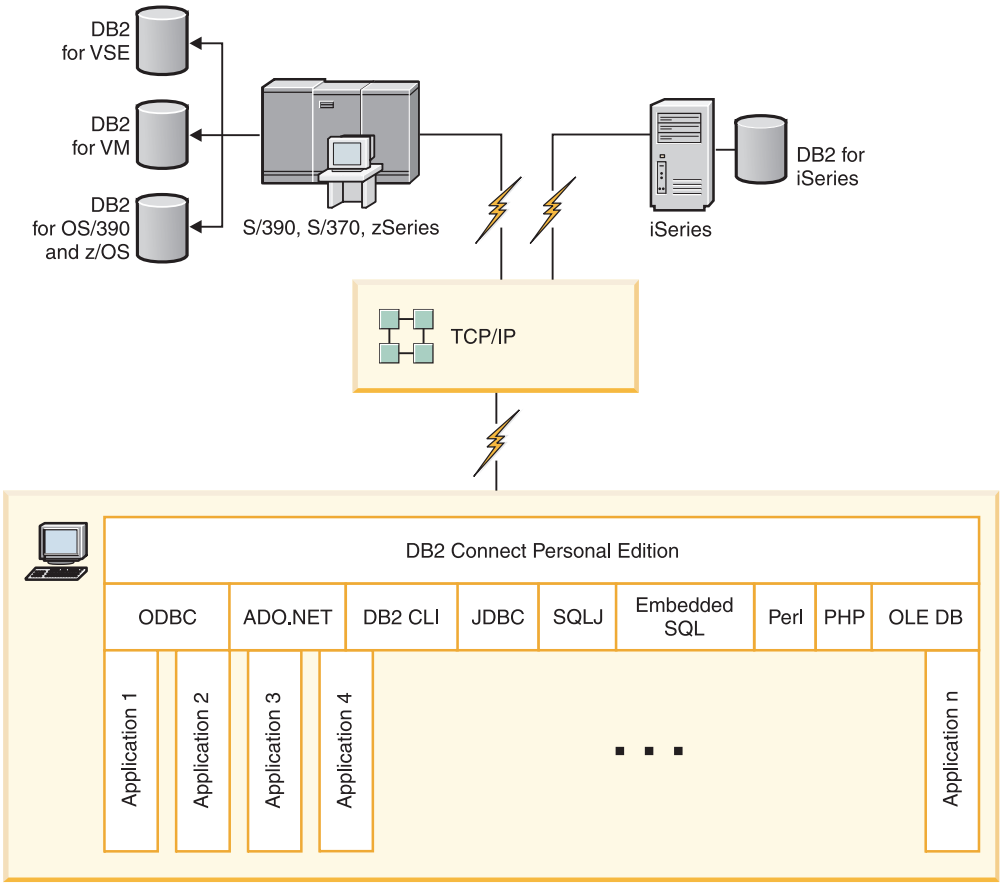


Figure 3. Direct Connection Between DB2 Connect and a host or System i database server

Note:

1. You do not need to have DB2 installed on the DB2 Connect workstation. If you want a complete relational database management system on the DB2 Connect workstation, order DB2.
2. The IBM Data Server 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 and z/OS.
3. 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 environment.
4. If a connection to a DB2 for z/OS database server with Sysplex exploitation enabled is lost, the client will automatically attempt to reestablish the connection.

Accessing host or System i DB2 data using DB2 Connect Personal Edition

A direct connection without intermediate servers is a very convenient and desirable configuration. This is especially true for situations where the host or System i database server supports TCP/IP connectivity. In such situations, each DB2 Connect workstation establishes a direct connection with the host or System i database server.

TCP/IP connectivity requires that the host or System i database support TCP/IP. The following versions support native TCP/IP connections:

- DB2 Universal Database (UDB) for OS/390 and z/OS Version 7.1 or later
- DB2 UDB for iSeries Version 5 Release 1 or later, and
- DB2 Server for VSE & VM Version 7 or later

To connect to an IBM host or System i database server you require a licensed DB2 Connect product. You cannot connect directly to an IBM host or System i Data Server using a IBM data server client.

Figure 4 shows a workstation, with DB2 Connect Personal Edition installed, directly connected to a host or System i database server.

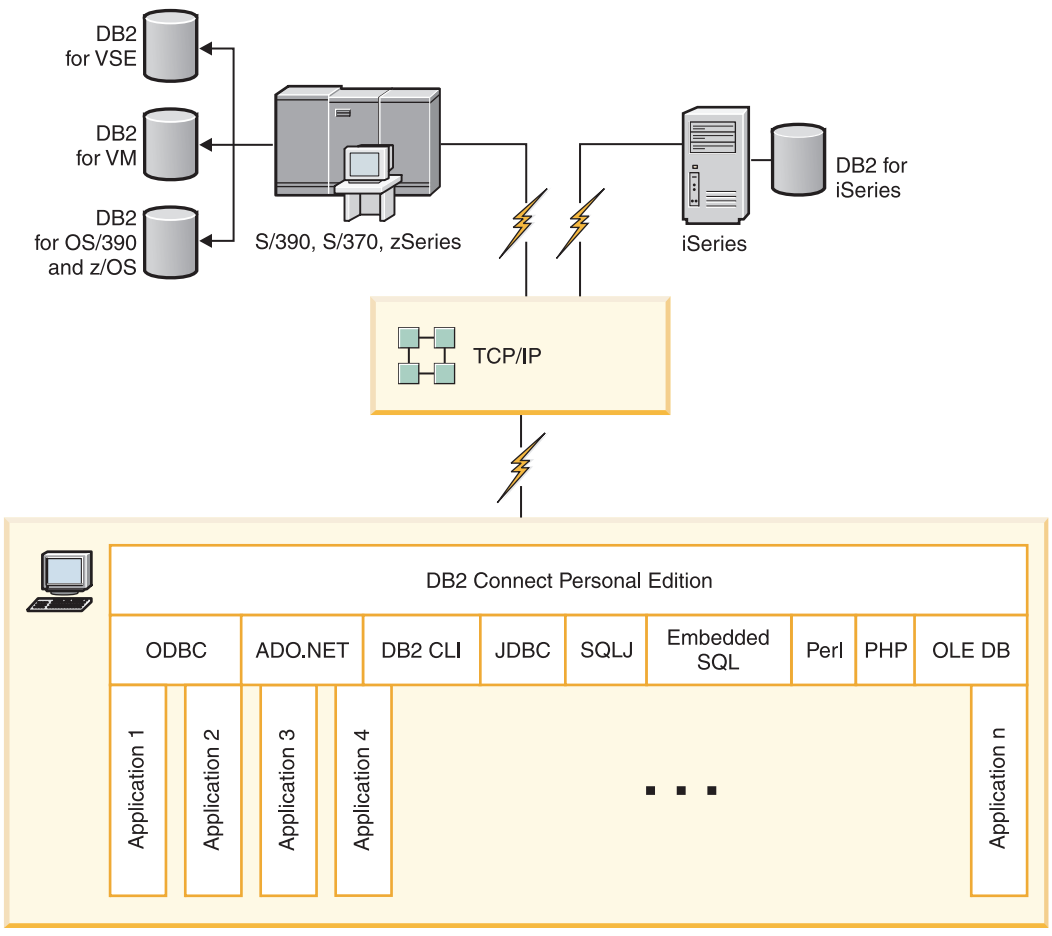


Figure 4. Direct connection between DB2 Connect and a host or System i database server

DB2 Connect server products as connectivity servers

A DB2 Connect server enables multiple clients to connect to host or System i data and can significantly reduce the effort that is required to establish and maintain access to enterprise data. Figure 5 illustrates IBM's solution for environments in which you want a DB2 client to make an indirect connection to a host or System i database server through a DB2 Connect server product, such as DB2 Connect Enterprise Server Edition.

Note: Indirect connections are supported only with DB2 clients or JCC clients running on Linux, UNIX, or Windows. Attempting to connect to a host or System i database server through a DB2 Connect server product using any other client results in an SQL1334 error.

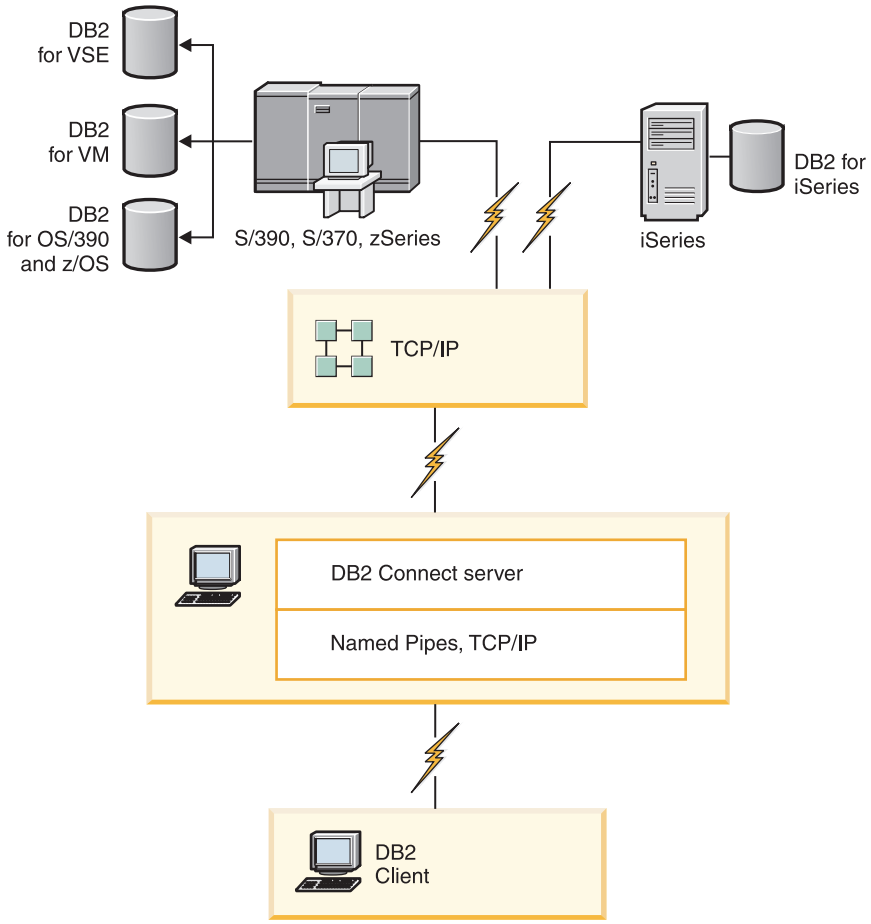


Figure 5. DB2 Connect Enterprise Server Edition

If a TCP/IP connection to the DB2 Connect server is lost, the client will automatically attempt to reestablish the connection. The client will first attempt to reestablish the connection to the original server. If the connection is not reestablished, the client will fail-over to an alternate DB2 Connect server. (The alternate server is specified on the server instance and its location is returned to the client during the connection.) If the connection to the alternate server is not reestablished, the client will attempt to reestablish the connection to the original server. The client will continue the attempts to reestablish the connection,

switching between the original server and the alternate server, until the connection is established or the number of attempts time out.

DB2 Connect and web applications

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

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. An example is an online catalog where 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 or PHP.

While CGI might 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, scalability can become an issue with large-scale e-commerce operations. Every time a CGI application is invoked, a new process is created on the web server. Each process must make its own connection to the database and submit 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 allow DB2 Connect users to reduce the number of queries being sent to the database.

Connection pooling reduces the frequency of connections and disconnections to and from a database.

Using PHP as a Web Server module or plug-in

Although PHP can be used for CGI programming, it is commonly used as a web server module or plug-in. In a multi-process web server such as Apache, the IBM DB2 driver for PHP can be used to mitigate the scalability issue. In a multi-process web server, a pool of processes are reused to service web server requests. To remove the need of building a database connection for every web request, a persistent connection can be created. In this environment, a persistent connection can exist beyond the scope of a single PHP script. The connection will be reused if an identical connection is needed by a subsequent web request.

DB2 Connect and IBM WebSphere

IBM WebSphere provides a more complete e-business solution than is possible with traditional scripting tools, such as PHP. WebSphere Application Servers not only performs the scripting possibilities of PHP, but also allow you to provide complex and high-end services through the web, using servlets, Active Server Pages, and enterprise JavaBeans™ and include support for Web-based technologies such as Java, TCP/IP, HTTP, HTTPS, HTML, DHTML, XML, MIME, SMTP, IIOP, and X.509, among others. 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 (AIX®, HP-UX, Linux, Novell NetWare, OS/390, z/OS, OS/400, Solaris operating system, Microsoft Windows)
- 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.

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.

WebSphere applications can be deployed on the same platform as the web server and DB2. In the case of the DB2 Universal Database (UDB) for OS/390 and z/OS, DB2 for z/OS, DB2 for VM, DB2 for VSE, DB2 UDB for iSeries, and DB2 for i5/OS, WebSphere is deployed on the same platform as the DB2 Connect server product.

There are several WebSphere solutions, as well as Rational® Application Developer (RAD). For more details, go to http://www.ibm.com/software/webservers/appserv/was/

DB2 Connect as a Java application server

Many of the shortcomings associated with scripting languages can be overcome by using Java instead. IBM provides both applets and applications that allow you to use Java 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 Perl DBI or 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 inline 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 and z/OS, DB2 for VSE and VM, DB2 Universal Database (UDB) for iSeries, and DB2 for i5/OS, 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 you can choose to go through a DB2 Connect server product that will provide connectivity to the host or the System i data server.

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 product and any industry standard Web server. If the web server and DB2 Connect are not on the same physical machines, a IBM data server client needs to be installed on the web server.

For DB2 for OS/390 and z/OS, the key component is a DB2 Connect server product running on a mid-tier server. This component provides JDBC server enablement, in addition to connecting to the DB2 for OS/390 and z/OS, DB2 for VSE and VM, DB2 Universal Database (UDB) for iSeries, and DB2 for i5/OS server. Again, there is no need for any special software for the client's web browser.

IBM provides extensive support and tooling for developing Java applications and applets. For database application development, DB2 Database Enterprise Developer Edition provides Rational Web Developer, IBM Data Studio, DB2 Embedded Application Server, Cloudscape™ Version 10.2, as well as DB2 and DB2 Connect for testing. Third-party tools such as NetBeans, Borland JBuilder or Symantec Visual Cafe will also work with IBM's database solutions.

DB2 Connect on the web server

IBM provides HTTP (Web) servers with all DB2 Connect products. DB2 Connect server products, such as DB2 Connect Enterprise Server Edition, provide 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 zSeries, System i, VM, and VSE systems, a DB2 Connect server product is required on the Web server. DB2 Connect server products will provide the libraries and communication interfaces to enable Web servers to access these host and System i platforms. TCP/IP can be used to communicate between the Web server and a database running on zSeries, System i, VM or VSE.

Note: IBM web solutions provide the ability to work with multiple databases within the same Common Gateway Interface (CGI) script (such as PHP) or within the same transaction in a CGI script.

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 Database for Linux, UNIX, and Windows, and DB2 Universal Database (UDB) on OS/390 and z/OS, DB2 for z/OS, DB2 UDB for iSeries, DB2 for i5/OS, and DB2 for VSE all share the same parameter convention for invoking stored procedures.

As with regular web interface scripts, the web browser submits the form to the web server, where the web interface 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 web interface 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.

DB2 Connect and 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. 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. DB2 Connect provides complete support for 2-tier client-server applications, where database servers are DB2 Universal Database (UDB) for OS/390 and z/OS, DB2 for z/OS, DB2 UDB for iSeries, DB2 for i5/OS, 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 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 server products 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.NET, DB2 CLI, Embedded SQL, JDBC, SQLJ, Perl, PHP, and OLE DB), 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 two-phase commit protocol support provided by DB2 Connect assures the integrity of such distributed transactions. For example, an application can update data in a DB2 for OS/390 and z/OS database and DB2 Database for Linux, UNIX, and Windows 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 a DB2 Connect server product, such as DB2 Connect Enterprise Server Edition.

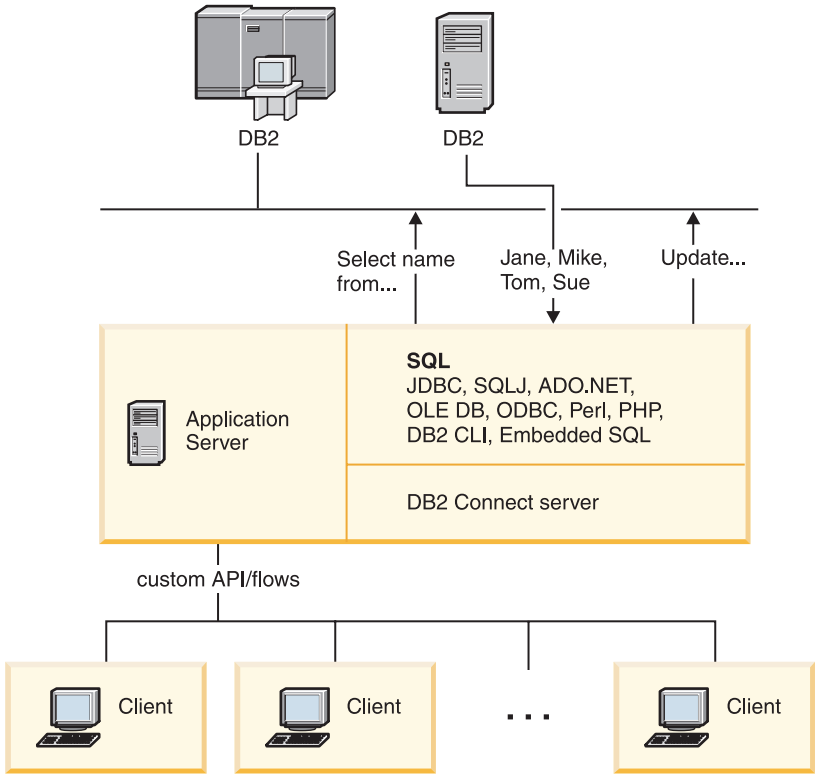


Figure 6. DB2 Connect support for application servers

Advanced features of DB2 Connect, such as connection pooling greatly reduce application resource requirements and simplify application server implementation.

DB2 Connect and application server configurations

A DB2 Connect server product 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. A DB2 Connect server product can be installed on either:

- The application server machine
- A separate communication server machine

In most situations, installing a copy of DB2 Connect on the same server as the application server 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 might 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 installing and maintaining an additional server.

Installing DB2 Connect on a separate server is a good option in situations where your DB2 Connect server product is not available for the operating system or hardware platform where application server is running.

DB2 Connect and transaction processing monitors

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.

Transaction processing

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 minimal. DB2 Connect allows applications under the control of an application server running on Linux, UNIX, and Windows to execute transactions against remote LAN, host, and System i database servers and have these transactions coordinated by a TP monitor.

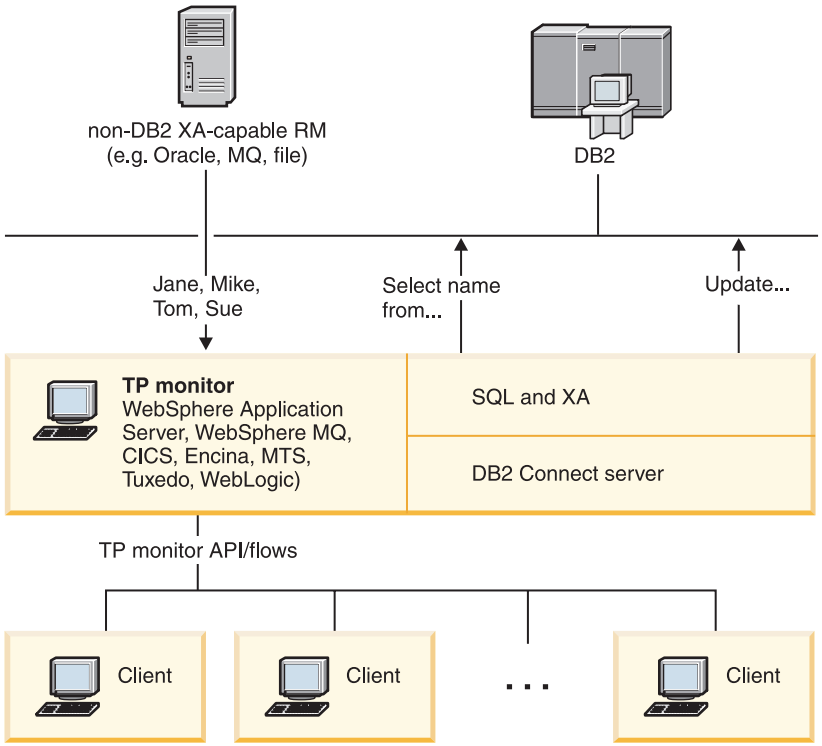


Figure 7. DB2 Connect support for TP monitors

In Figure 7, the APIs, as well as the connectivity mechanism between the application server and the back-end database servers, are provided by a DB2 Connect server product, such as DB2 Connect Enterprise Server Edition.

Examples of transaction processing monitors

The most common TP monitors on the market today are:

- IBM WebSphere Application Server
- IBM WebSphere MQ
- IBM TxSeries CICS®
- IBM TxSeries Encina® Monitor
- BEA Tuxedo
- BEA WebLogic
- Microsoft Transaction Server (MTS)

Remote System i, zSeries, and LAN database servers can be used within transactions coordinated by these TP monitors.

X/Open Distributed Transaction Processing (DTP) model

An application executing business logic might 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 z/OS 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, System i, and DB2 LAN-based databases, when accessed via DB2 Connect, are XA compliant RMs. Therefore, any TP monitor which has an XA compliant TM can use host, System i, and LAN-based DB2 databases within business applications executing transactions.

Part 2. DB2 Connect reference

Chapter 4. Updating database directories

DB2 Connect uses the following directories to manage database connection information:

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

Note:

1. Before updating these directories, you should configure communications on the host or System i database server and workstations.
2. Database directories can be updated using the Configuration Assistant (CA).

To update database directories:

1. Collect database directory information using the directory customization worksheet
2. Refer to the "Update the directories with information about remote database server machines" topic in the Control Center

System database directory values

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 System i 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 made for connections originating from the DB2 Connect server. The valid options are: `SERVER`, `SERVER_ENCRYPT`, `CLIENT`, `KERBEROS`, and `DATA_ENCRYPT`. There is no support for the GSSPLUGIN authentication type in the system database directory.

Node directory values

You can specify the following information in the node directory:

Node name

A nickname for the host or System i 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

Must be `TCP/IP`.

Security type

The type of security checking that will be done. 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.

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 and z/OS remote hosts, the hostname appears in the DSNL004I message (DOMAIN=hostname) when the Distributed Data Facility (DDF) is started. The -DISplay DDF command could also be used.

If accessing a z/OS data sharing group, the domain name should map to the DB2 group dynamic VIPA address. This address routes to the least loaded DB2 member. To access a specific member use the specific DB2 member dynamic VIPA address and turn off sysplex routing. Each member DSNL004I message displays the member specific domain name.

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 and z/OS 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. The -DISplay DDF command could also be used.

If accessing a z/OS data sharing group, the domain name should map to the DB2 group dynamic VIPA address. This address routes to the least loaded DB2 member. To access a specific member use the specific DB2 member dynamic VIPA address and turn off sysplex routing. Each member DSNL004I message displays the member specific domain name.

Note: A second port used for two-phase commit resynchronization operations over TCP/IP connections can be assigned by the server. For example, the DB2 Universal Database for z/OS and OS/390 bootstrap data set assigns a port number (RESPORT) to be used for resynchronization for inbound connections to DB2 Universal Database for z/OS and OS/390 only. No service name need be defined for this.

DCS directory values

You can specify the following information in the DCS directory:

Database name
A user-defined nickname for the host or System i 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 System i database server system, as follows:

OS/390 and z/OS
A DB2 Universal Database for z/OS and OS/390 subsystem identified by its LOCATION NAME or one of the alias LOCATION names defined on the z/OS server.

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
```

multiple LOCATION NAMEs are 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. The -DISplay DDF command could also be used.

If accessing a z/OS data sharing group, the domain name should map to the DB2 group dynamic VIPA address. This address routes to the least loaded DB2 member. To access a specific member use the specific DB2 member dynamic VIPA address and turn off sysplex routing. Each member DSNL004I message displays the member specific domain name.

VSE or VM
The database name (DBNAME)

OS/400 and z/OS
The relational database name (RDBNAME)

Other For Windows, Linux, and UNIX operating systems, the database alias found in the database directory.

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

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

Note: When processing a query request, the DRDA server returns data in the form of a set of rows that represent the result set. With each row, there is also an SQLCA returned, usually containing a zero or positive sqlcode (such as +12

or +802). If you use a customized mapping file at a DB2 Connect server, such positive sqlcodes will not be mapped if they are contained in the customized mapping file and have customized mappings (for example, they are mapped to a different sqlcode or have customized token mappings).

It is important to emphasize that:

1. Positive sqlcodes represent warnings, as opposed to negative sqlcodes which indicate error conditions. All the negative sqlcodes will always be mapped in all circumstances, regardless of which mapping file is being used. All the positive sqlcodes, contained in the customized mapping file and mapped to themselves with no change, will always be mapped as well. Also, those positive sqlcodes that are not contained in the customized mapping file at the DB2 Connect server will also always be mapped.
2. If you use the default mapping file, or you connect to the host database directly, the sqlcode mapping will always be performed for all sqlcodes.

,D This is the second positional parameter. If it is specified the application will disconnect from the host or System i 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. INTERRUPT_ENABLED only applies if the end server does not support interrupts. If a server supports the DRDA interrupt flow DB2 Connect will simply pass the interrupt request on to the server.

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 System i 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, and Windows.

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 System i database server, in order to process additional database requests. On platforms other than AIX V5.2 and later and Windows, 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 might kill the socket, but - depending on the server implementation - there might or might not be an outstanding receive. DB2 Universal Database for z/OS and OS/390 uses 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.

,,,,,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 TCPIP node nynode remote myhost server myport
catalog dcs database nydb1 as new_york
catalog database nydb1 as newyork1 at node nynode
     authentication server
```

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 quotation marks, the LOCALDATE value has to be specified inside two pairs of double quotation marks. Note the use of the operating system escape character "\" (backslash) to ensure that the double quotation marks are not stripped from the LOCALDATE specification.

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

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.

,,,,,,,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.

Directory customization worksheet

The directory customization worksheet shows the information that you need to collect. You might find it convenient to make a copy of the worksheet and enter your system values.

Node Directory Parameters

Table 1. Node Directory Parameters

Parameter	Example	Your value
Node name	DB2NODE	
Remote hostname (TCP/IP node)	ZOSHOST	
Server (TCP/IP service name or port number)	db2inst1c (or 446)	

Note:

1. The default TCP/IP port number for DRDA is 446
2. Unless you know that the host or System i database server supports SECURITY SOCKS, do not specify SECURITY for a TCP/IP node.

DCS Directory Parameters

Table 2. DCS Directory Parameters

Parameter	Example	Your value
Database name	DB2DB	
Target database name	NEW_YORK3	
Application requester		
Parameter string	″,,,,,,LOCALDATE=\″\″YYMMDD\″\″\″	

System Database Directory Parameters

Table 3. System Database Directory Parameters

Parameter	Example	Your value
Database name	DB2DB	
Database alias	NYC3	
Node name	DB2NODE	
Authentication	SERVER	

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 System i 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.

Handling BiDi data

The following section applies to OS/390 and z/OS servers only. This feature must not be enabled for a DB2 for i5/OS server as full BiDi support is already provided.

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

- Numeral shape (ARABIC versus HINDI)
- Orientation (RIGHT-TO-LEFT versus LEFT-TO-RIGHT)
- Shaping (SHAPED versus UNSHAPED)
- Symmetric swapping (YES or NO)
- Text type (LOGICAL versus 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 OS/390 or z/OS data is usually in SHAPED VISUAL format. Therefore, without any support for BiDi attributes, data sent from DB2 for OS/390 and z/OS 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. 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 IBM data server 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 62245 (BiDi string type 10).

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 62245. The second problem is that the DB2 host database does not recognize the IBM data server client CCSID of 62213. It only supports CCSID 62209 (BiDi string type 10), 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=62245"
```

This tells DB2 Connect to override the DB2 host database CCSID of 424 with 62245. This override includes the following processing:

1. DB2 Connect will connect to the DB2 host database using CCSID 62209 (BiDi string type 10).
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 62209 (BiDi string type 10).
3. DB2 Connect will perform BiDi layout transformation on data it receives from the DB2 host database from CCSID 62245 (BiDi string type 10) to CCSID 62213 (BiDi string type 5).

Note:

1. The environment variable or registry value DB2BIDI has to be set to YES in order for the BIDI parameter to take effect. DB2BIDI must be set at the DB2 Connect workstation where the DCS database directory entry is catalogued. For applications running on a client remote to a DB2 Connect server, the DB2BIDI variable must be set at that client as well.
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 might 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.

Parameter string specifications

The following are examples of DCS parameters (each line is a set of parameters):

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

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

Note: You must use the operating system escape character "\" (backslash) when using CLP from the operating system's command line on UNIX systems because of the need to specify two pairs of double quotation marks when specifying the LOCALDATE mask in the parameter string. 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
```

Chapter 5. DB2 Connect security

Authentication of users is important when using DB2 Connect because the users may be both local or remote to DB2 Connect and to the database that has the data they wish to access. Trusted connections and Kerberos support are presented along with security considerations for databases on host machines.

Trusted connections through DB2 Connect

Some DB2 database servers support trusted contexts. A *trusted context* allows the database administrator to, among other things, define conditions under which a client application will be allowed to create a trusted connection. A *trusted connection* is allowed to do things that a normal connection cannot.

There are two types of trusted connection, implicit and explicit. When you create a connection, whether you get an explicit trusted connection, an implicit trusted connection, or a regular connection depends on whether you ask for a trusted connection and whether the connection meets the criteria defined in the trusted context on the server, as summarized in Table 4.

Table 4. What type of connections result from different combinations of actions

	The connection meets the server's criteria for being trusted	**The connection does not meet the server's criteria for being trusted**
You request that the connection be trusted	Explicit trusted connection	Regular connection and warning SQL20360W (SQLSTATE 01679) is returned.
You do not request that the connection be trusted	Implicit trusted connection	Regular connection

An *implicit trusted connection* is identical to a regular connection except that it grants temporary role privileges to the user while they are using the connection. The role privileges that are granted (if any) are specified in the trusted context that caused the connection to be trusted.

Implicit trusted connections can be created by any application that connects using DB2 Connect. Implicit trusted connections are made and used in the same way that regular connections are made and used. This means that no code changes are necessary for an existing application to take advantage of implicit trusted connections as long as the application connects through DB2 Connect.

An *explicit trusted connection* grants temporary role privileges to the user the same way that an implicit trusted connection does. In addition, an explicit trusted connection lets you change the authorization ID used when performing actions across that connection. Changing the authorization ID on an explicit trusted connection is referred to as *switching users*. The authorization IDs to which you can switch and whether a given authorization ID requires a password when switching to it are defined as part of the trusted context that allowed the trusted connection to be created.

User switching can significantly reduce the overhead of sharing a connection among several users, especially for user names that do not require a password because in that case the database server does not authenticate the authorization ID. When using the feature, however, you must be very certain that your application does not allow switching to an authorization ID without validating and authenticating that authorization ID. Otherwise you are creating a security hole in your system.

Explicit trusted connections can be created and the user can be switched when connecting through DB2 Connect using CLI or JDBC, including XA established connections. Creating an explicit trusted connection and switching users requires setting special connection attributes. This means that existing applications will need to be modified in order to take advantage of explicit trusted connections.

Other than the differences just mentioned, you can use a trusted connection (whether implicit or explicit) the same way you would used a regular connection. You must be certain, however, to explicitly disconnect an explicit trusted connection when you are done with it, even if it is in a broken or disconnected state. Otherwise resources used by the connection might not be released. This is not a problem with implicit trusted connections.

Note:

1.

   Important: Switching users without supplying a password bypasses the database server's authentication. Your application must not allow a switch to an authorization ID without a password unless that application has already validated and authenticated that authorization ID. To do otherwise creates a security hole.
2. Explicit trusted connections should not use CLIENT authentication. This does not apply to implicit trusted connections.
3. Applications using explicit trusted connections should be run on secure machines which are password protected and accessible only to authorized personnel. This does not apply to implicit trusted connections.

Creating and terminating a trusted connection through CLI

If the database server you are connecting to is configured to allow it, you can create an explicit trusted connection when connecting through CLI.

This procedure assumes that you are not using an XA transaction manager. If you are using an XA transaction manager you only need to make sure that the transaction manager is configured to set the configuration value TCTX to TRUE when it calls xa_open. If that is done then any connection that can be an explicit trusted connection will be. To verify that a connection is an explicit trusted connection see step 3.

- The database that you are connecting to must support trusted contexts.
- A trusted context must be defined that will recognize the client as being trustable.
- You must know the system authorization ID that is specified in the trusted context. The system authorization ID of a trusted connection is the authorization ID you provide to the server as a user name when creating the connection. For your connection to be trusted by a particular trusted context the system

authorization ID must be the one specified in that trusted context. Ask your security administrator for a valid system authorization ID and the password for that ID.

The examples in these instructions use the C language and assume that conn is a pointer to a valid, but unconnected, connection handle. The variable rc is assumed to have a data type of SQLRETURN.

1. In addition to setting any connection attributes that you would set for a regular connection, set the connection attribute SQL_ATTR_USE_TRUSTED_CONTEXT to SQL_TRUE with a call to the SQLSetConnectAttr function.

```
rc = SQLSetConnectAttr(
    conn,
    SQL_ATTR_USE_TRUSTED_CONTEXT, SQL_TRUE, SQL_IS_INTEGER
);
```

2. Connect to the database as you would for a regular connection, by calling the SQLConnect function for instance. Use the system authorization ID as the user name and its password as the password. Be sure to check for errors and warnings, especially those listed in table Table 5.

Table 5. Errors indicating failure to create a trusted connection

SQLCODE	SQLSTATE	Meaning
SQL20360W	01679	The connection could not be established as a trusted connection. It was established as a regular connection instead.

   If no errors or warnings tell you differently, then the connection is established and is an explicit trusted connection.
3. (Optional) You can verify that an established connection is an explicit trusted connection by checking the value of the connection attribute SQL_ATTR_USE_TRUSTED_CONTEXT using the SQLGetConnectAttr function. If it is set to SQL_TRUE the connection is an explicit trusted connection.
4. When you are finished using the connection you must be very careful to explicitly disconnect it, even if it is in a broken or disconnected state. If you do not explicitly disconnect an explicit trusted connection some of the resources used by the connection might not be released.

Note:

1. Explicit trusted connections should not use CLIENT authentication. This does not apply to implicit trusted connections.
2. Applications using explicit trusted connections should only be run on secure computers which are password protected and accessible only to authorized personnel. This does not apply to implicit trusted connections.

Switching users on a trusted connection through CLI

You can switch users on an explicit trusted connection through the command line interface (CLI). For a description of what it means to switch users using a trusted connection see the topic in the related links.

- The connection must have been successfully created as an explicit trusted connection.
- The explicit trusted connection must not be in a transaction.
- The trusted context that allowed the explicit trusted connection to be created must be configured to allow switching to the authorization ID you are switching to.

The examples in these instructions use the C language and assume that **conn** is a pointer to a connected explicit trusted connection. The variable *rc* is assumed to have a data type of SQLRETURN. The variable *newuser* is assumed to be a pointer to a character string holding the authorization ID of the user you want to switch to. The variable *passwd* is assumed to be a pointer to a character string containing the password for that authorization ID.

1. Call the SQLSetConnectAttr function to set the SQL_ATTR_TRUSTED_CONTEXT_USERID attribute. Set it to the authorization ID you want to switch to.

   ```
   rc = SQLSetConnectAttr(
       conn,
       SQL_ATTR_TRUSTED_CONTEXT_USERID, newuser, SQL_NTS
   );
   //Check for errors
   ```

   Be sure to check for errors and warnings, especially those listed in table Table 6.

Table 6. Errors indicating failure to set a new authorization ID when switching users

SQLCODE	Meaning
CLI0106E	The connection is not connected.
CLI0197E	The connection is not a trusted connection.
CLI0124E	There is a problem with the value provided. Check that it is not null, or not too long, for example.
CLI0196E	The connection is involved in a unit of work that prevents it from switching users. To be able to switch users the connection must not be in a transaction.

2. (Optional unless the trusted context that allowed this trusted connection requires a password for the authorization ID you are switching to) Call the SQLSetConnectAttr function to set the SQL_ATTR_TRUSTED_CONTEXT_PASSWORD attribute. Set it to the password for the new authorization ID.

   ```
   rc = SQLSetConnectAttr(
       conn,
       SQL_ATTR_TRUSTED_CONTEXT_PASSWORD, passwd, SQL_NTS
   );
   //Check for errors
   ```

   Be sure to check for errors and warnings, both those listed in table Table 6 and those listed in table Table 7.

Table 7. Errors indicating failure to set a password when switching users

SQLCODE	Meaning
CLI0198E	The attribute SQL_ATTR_TRUSTED_CONTEXT_USERID has not yet been set.

3. Proceed as with a regular connection. If you are using an XA transaction manager the user switch is attempted as part of the next request, otherwise the user switch is attempted just prior to initiating the next function call that accesses the database (SQLExecDirect for example). In either case, in addition to the errors and warnings you would normally check for, be sure to check for the errors listed in Table 8 on page 43. The errors in Table 8 on page 43 indicate that the user switch failed.

Table 8. Errors indicating failure to switch users

SQLCODE	Meaning
SQL1046N	The trusted context that allowed this trusted connection is not configured to allow switching to the authorization ID you are trying to switch to. You will not be able to switch to that authorization ID until the trusted context is changed.
SQL30082N	The password provided is not correct for the authorization ID you are switching to.
SQL0969N with a native error of -20361	There is some database level constraint that prevent you from switching to the user.

If the user switch fails the connection will be in an unconnected state until you successfully switch to another user. You can switch users on a trusted connection in an unconnected state but cannot access the database server with it. A connection in an unconnected state will remain in that state until you successfully switch users on it.

Notes:

1. **Important:** Switching users without supplying a password bypasses the database server's authentication. Your application must not allow a switch to an authorization ID without a password unless that application has already validated and authenticated that authorization ID. To do otherwise creates a security hole.
2. Specifying a NULL value for the SQL_ATTR_TRUSTED_CONTEXT_USERID attribute is equivalent to specifying the trusted context system authorization ID (the user id used when the explicit trusted connection was created).
3. When you successfully set the value of the SQL_ATTR_TRUSTED_CONTEXT_USERID connection attribute on an explicit trusted connection the connection is immediately reset. The result of resetting is as if a new connection were created using the original connection attributes of that connection. This reset happens even if the value you set the connection attribute to is the system authorization ID or NULL or the same value that the attribute currently holds.
4. If the SQL_ATTR_TRUSTED_CONTEXT_PASSWORD attribute is set, the password will be authenticated during the switch user processing, even if the trusted context that allowed the trusted connection doesn't require authentication on a switch user for that authorization ID. This results in unnecessary overhead. This rule doesn't apply to the trusted context system authorization ID. If the trusted context system authorization ID doesn't require authentication when you switch to it then it is not authenticated even if a password is provided.

DB2 Connect authentication considerations

As a DB2 Connect administrator, in cooperation with your host or System i database administrator, you can determine where user names and passwords are validated:

- At the client
- At the host or System i server
- Single sign-on and validation through a third-party system (Kerberos).

Note: If the remote client has not specified an authentication type, the client will default to SERVER_ENCRYPT. If this type is not accepted by the server, the client will attempt to retry using an appropriate value returned from the server. To help optimize performance, always specify the authentication type at the client to avoid this extra network flow.

Starting with DB2 Connect Version 8.2.2 (equivalent to Version 8.1 FixPak 9) the gateway is no longer a passive participant during authentication negotiation. Instead, the gateway takes an active role. The authentication type specified in the database directory entry at the gateway overrides the authentication type cataloged at the client. The client, gateway, and server must all specify compatible types. If the cataloged authentication type at the gateway has not been specified in the database directory entry, SERVER authentication will be the default type requested of the server. However, negotiation will still take place between the client and server if the server does not support SERVER authentication. This behavior is in contrast to the client which defaults to SERVER_ENCRYPT if an authentication type has not been specified.

The authentication type cataloged at the gateway is not used if DB2NODE or the SQL_CONNECT_NODE option of the Set Client API has been set at the client. In these cases negotiation is still strictly between the client and the server.

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 host or System i server database.

SERVER_ENCRYPT
As for SERVER authentication, the user name and password are validated at the host or System i database server, but the transferred passwords are encrypted at the client.

DATA_ENCRYPT
Provides the ability to encrypt user data during client/server communications.

KERBEROS
Enables the client to log into the server using Kerberos authentication instead of the traditional ID and password combination. This authentication type requires that both the server and client be Kerberos-enabled.

Kerberos authentication is unique in that the client does not pass a user ID and password directly to the server. Instead, Kerberos acts as a third-party authentication mechanism. The user enters an ID and password once at the client terminal, and Kerberos validates this sign-on. After this, Kerberos automatically and securely passes the user's authorization to any local and network services requested. This means that the user does not need to re-enter an ID and password to log into a remote DB2 server. The single sign-on capability provided by Kerberos authentication requires that both DB2 Connect and the database server that it is connecting to provide Kerberos support.

Note: There is no support for the GSSPLUGIN authentication type.

Kerberos support

The Kerberos authentication layer which handles the ticketing system is integrated into the Windows 2000 Active Directory mechanism. The client and server sides of an application communicate with the Kerberos SSP (Security Support Provider) client and server modules respectively. The Security Support Provider Interface (SSPI) provides a high level interface to the Kerberos SSP and other security protocols.

Typical setup

To configure DB2 with Kerberos authentication, set up:

- An authorization policy for DB2 (as a service) in the Active Directory that is shared on a network, and
- A trust relationship between Kerberos Key Distribution Centers (KDCs)

In the simplest scenario, there is at least one KDC trust relationship to configure, that is, the one between the KDC controlling the client workstation, and the System i, OS/390 or z/OS system. OS/390 Version 2 Release 10 or z/OS Version 1 Release 2 provides Kerberos ticket processing through its RACF® facility which allows the host to act as an UNIX KDC.

DB2 Connect provides as usual the router functionality in the 3-tier setting. It does not assume any role in authentication when Kerberos security is used. Instead, it merely passes the client's security token to DB2 for i5/OS or to DB2 for OS/390 and z/OS. There is no need for the DB2 Connect gateway to be a member of the client or the host's Kerberos realm.

Downlevel compatibility

DB2 minimum requirements for Kerberos support:

IBM data server client:
Version 8

DB2 Connect:
Version 8

DB2 Universal Database (UDB) for OS/390 and z/OS:
Version 7

Hints and tips about OS/390 and z/OS security

This topic provides some hints and tips about security for DB2 Connect connecting to a DB2 for OS/390 and z/OS database server.

Extended security field

Ensure that the DB2 OS/390 and z/OS Extended Security Field is set to `YES`. This field appears in the DB2 for OS/390 and z/OS DSNTIPR panel.

Extended security codes

Until DB2 Universal Database for z/OS and 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 z/OS and 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 z/OS and OS/390 ZPARM installation parameter for extended security should be set to the value YES. Use the DB2 Universal Database for z/OS and 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, Windows, Linux, UNIX, and Web applications using DB2 Connect will receive an SQL30082 error message.

TCP/IP security already verified

If you want to provide support for the DB2 security option AUTHENTICATION=CLIENT, then use DB2 Universal Database for z/OS and 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 might create security concerns in some installations. DB2 Universal Database for z/OS and OS/390 introduces a new bind option DYNAMICRULES(BIND) that allows execution of dynamic SQL under the authorization of either the owner or the binder.

DB2 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.

Password change support

If a user ID's password has expired, an SQL CONNECT statement returns an error message, such as SQLCODE -30082 reason code 1. With DB2 Connect it is possible to change the password remotely. Through DRDA, DB2 Universal Database for z/OS and OS/390 can change the password for you, by issuing the following CONNECT statement:

```
CONNECT TO <database> USER <userid> USING <password>
   NEW <new_password> CONFIRM <new_password>
```

The "Change password" dialog of the DB2 Configuration Assistant can also be used to change the password.

Security types supported with DB2 Connect

This topic lists the various combinations of authentication and security settings that are supported with DB2 Connect.

Security types for TCP/IP connections

The TCP/IP communication protocol does not support security options at the network protocol layer. The authentication type determines where authentication takes place. Only the combinations shown in this table are supported by DB2 Connect. The authentication setting is in the database

directory entry at the DB2 Connect server.

Table 9. Valid Security Scenarios

Scenario	Authentication setting	Validation
1	CLIENT	Client
2	SERVER	Host or System i database server
3	SERVER_ENCRYPT	Host or System i database server
4	KERBEROS	Kerberos security
5	DATA_ENCRYPT	Host or System i database server

Discussion of security types

The following discussion applies to the connections described above and listed in Table 9. Each scenario is described in more detail, as follows:

- In scenario 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 they 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 that can be trusted.
- In scenario 2, the user name and password are validated at the host or System i database server only. The user ID and 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 System i database server.
- Scenario 3 is the same as scenario 2, except that the user ID and password are encrypted.
- In scenario 4, a Kerberos ticket is obtained by the client from the Kerberos KDC. The ticket is passed unaltered through DB2 Connect to the server, where it is validated by the server.
- Scenario 5 is the same as scenario 3, except that the user data is also encrypted.

Chapter 6. Binding applications and utilities (DB2 Connect)

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 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 a process of processing these bind files by a host or System i database server.

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

- ddcsmvs.lst for OS/390 or z/OS
- 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 a DB2 Connect server product is installed, the DB2 Connect utilities must be bound to each host or System i database server before they can be used with that system. Assuming the clients are at the same fix pack level, you need to bind the utilities only once, regardless of the number of client platforms involved.

For example, if you have 10 Windows clients, and 10 AIX clients connecting to DB2 Universal Database (UDB) for OS/390 and z/OS via DB2 Connect Enterprise Server Edition on a Windows server, do one the following:

- Bind ddcsmvs.lst from one of the Windows clients.
- Bind ddcsmvs.lst from one of the AIX clients.
- Bind ddcsmvs.lst from the DB2 Connect server.

This example assumes that:

- All the clients are at the same service level. If they are not then, in addition, you might need to bind from each client of a particular service level
- The server is at the same service level as the clients. If it is not, then you need to bind from the server as well.

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 System i database server that you are binding to, do the following:

1. Make sure that you have sufficient authority for your host or System i database server management system:

   OS/390 or z/OS

   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 System i database server, ddcsmvs.lst is the bind list file for z/OS, and *path* represents the location of the bind list file.

For example *drive*:\sqllib\bnd\ applies to all Windows operating systems, and *INSTHOME*/sqllib/bnd/ applies to all Linux and 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
```

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
```

Note:

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 (CLP). 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 might 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 System i database server should be expected to produce some warnings.

b. If you are connecting to a DB2 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 System i database server. Also, to connect to a DB2 database, it is recommended 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 might need to bind the utilities on these clients to DB2 Connect.

Chapter 7. Multisite Updates

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 processing monitors (TP monitors) 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 a distributed unit of work (DUOW). Currently, the following DB2 servers provided DUOW support that enabled them to participate in distributed transactions:

- DB2 for Linux, UNIX and Windows Version 8 or later
- DB2 Universal Database (UDB) for OS/390 and z/OS Version 7
- DB2 for z/OS Version 8
- DB2 for i5/OS requires OS/400 Version 5 Release 1 or later

A distributed transaction can update any mix of supported database servers. For example, your application can update several tables in a DB2 database on Windows, a DB2 for OS/390 and z/OS database, and a DB2 for i5/OS database, all within a single transaction.

Enabling Multisite Updates using the Control Center

You can use the Control Center to provide multisite updates.

To enable multisite updates:

1. Launch the Control Center.
2. Click the [+] sign to expand the tree view.
3. With the right mouse button, select the instance that you want to configure. A pop-up menu opens.
4. Select **Multisite Update —> Configure** menu item. The Multisite Update Wizard opens.
5. Select **Use the TP monitor named below** and 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**.

   Click **Next**.
6. If you are using a TP monitor, specify the sync point manager settings. If you are not using a TP monitor, specify your transaction manager database.
7. Click **Finish**.

Testing Multisite Update using the Control Center

You can test your multisite update setup using the Control center.

To test multisite update:

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.
2. Select the databases you want to test from the available databases in the **Available** list box. You can use the arrow buttons (> and >>) in the middle to move selections to and from the **Selected** list box. You can also change the selected userid and password by directly editing them in the **Selected** list box.
3. When you have finished your selection, click **OK**. The Multisite Update Test Result window opens.
4. The Multisite Update Test Result window shows which of the databases you selected have succeeded or failed the update test. The window will show SQL codes and error messages for those that failed. Click **Close** to close the window.
5. Click **Close** to close the Test Multisite Update window.

Multisite update and sync point manager

Host and System i database servers require DB2 Connect to participate in a distributed transaction originating from Linux, Windows, UNIX, and web applications. In addition, many of the multisite update scenarios that involve host and System i database servers require that the sync point 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 (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 if DB2 Connect is required for any access to the host or System i from Intel® or UNIX machines. For multisite updates, the SPM component of DB2 Connect is required if you are using a TP monitor.

Table 10. Multisite update scenarios that require SPM – TCP/IP

Transaction Processor Monitor Used?	Sync Point Manager Needed?	Product Required (Choose One)	Host and System i Database Supported
Yes	Yes	DB2 Connect server product DB2 Enterprise Server Edition with DB2 Connect license applied	DB2 Universal Database (UDB) for OS/390 and z/OS V7 DB2 UDB for z/OS V8 or later
No	No	DB2 Connect Personal Edition DB2 Connect server product DB2 Enterprise Server Edition with DB2 Connect license applied	DB2 UDB for OS/390 and z/OS V7 DB2 UDB for z/OS V8 or later

Note: A distributed transaction can update any mix of supported database servers. For example, your application can update several tables in a DB2 database on Windows, a DB2 for OS/390 database and a DB2 for i5/OS database all within a single transaction.

Configuring DB2 Connect with an XA compliant transaction manager

This topic describes the configuration steps necessary to use S/390, System i, and zSeries database servers within your TP monitor.

You must have an operational TP monitor and have DB2 Connect installed, as well as have configured and tested a connection to the host or System i database server.

To configure DB2 Connect to use S/390, System i, and zSeries 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.
2. Configure the TP monitor with DB2's XA_OPEN string. Each TP monitor has its own way to do this. 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 System i (Version 5 Release 3 and earlier)

database servers do not yet support the XA interface. System i Version 5 Release 4 and following has full XA support.

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 System i database servers. By default, the DB2 instance has predefined 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 and z/OS, 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.

DB2 Connect support for loosely coupled transactions

The support within DB2 Connect for loosely coupled transactions is intended for users who implement XA distributed applications that access DB2 Universal Database (UDB) for i5/OS Version 5 Release 4 or later; and DB2 UDB for OS/390 and z/OS Version 7 or later. This support allows different branches of the same global transaction to share lock space on DB2 UDB for OS/390 and z/OS.

Support for loosely coupled transactions is intended for .NET and COM+ applications.

This feature reduces the window where one branch of a distributed transaction encounters lock timeout or deadlock as a result of another branch within the same global transaction.

Chapter 8. Moving data with DB2 Connect

If you are working in a complex environment in which you need to move data between a host database system and a workstation, you can use DB2 Connect, the gateway for data transfer between the host and the workstation (see Figure 8).

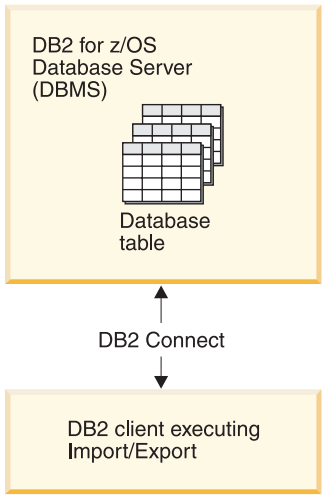


Figure 8. Import/Export through DB2 Connect

The DB2 export and import utilities allow you to move data from a host or System i server database to a file on the DB2 Connect workstation, and the reverse. You can then use the data with any other application or relational database management system that supports this export or import format. For example, you can export data from a host or System i server database into a PC/IXF file, and then import it into a DB2 for Windows database.

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

Note:

1. The data to be exported or imported must comply with the size and data type restrictions that are applicable to both databases.
2. To improve import performance, you can use compound queries. Specify the `compound` file type modifier in the import utility to group a specified number of query statements into a block. This can reduce network overhead and improve response time.

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

- The file type must be PC/IXF.
- A target table with attributes that are compatible with the data must be created on the target server before you can import to it. The db2look utility can be used to get the attributes of the source table. Import through DB2 Connect cannot create a table, because INSERT is the only supported option.

If any of these conditions is not met, the operation fails, and an error message is returned.

Note: Index definitions are not stored on export or used on import.

If you export or import 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 the 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 column data has a consistent pattern.

Moving Data from a workstation to a host server

To move data to a host or System i server database:

1. Export the data from a DB2 table to a PC/IXF file
2. Using the INSERT option, import the PC/IXF file into a compatible table in the host server database.

To move data from a host server database to a workstation:

1. Export the data from the host server database table to a PC/IXF file.
2. Import the PC/IXF file into a DB2 table.

Example

The following example illustrates how to move data from a workstation to a host or System i server database.

Export the data into an external IXF format by issuing the following command:

```
db2 export to staff.ixf of ixf select * from userid.staff
```

Issue the following command to establish a DRDA connection to the target DB2 database:

```
db2 connect to cbc664 user admin using xxx
```

If it doesn't already exit, create the target table on the target DB2 database instance:

```
CREATE TABLE mydb.staff (ID SMALLINT NOT NULL, NAME VARCHAR(9),
       DEPT SMALLINT, JOB CHAR(5), YEARS SMALLINT, SALARY DECIMAL(7,2),
       COMM DECIMAL(7,2))
```

To import the data issue the following command:

```
db2 import from staff.ixf of ixf insert into mydb.staff
```

Each row of data will be read from the file in IXF format, and an SQL INSERT statement will be issued to insert the row into table mydb.staff. Single rows will continue to be inserted until all of the data has been moved to the target table.

Detailed information is available in ″Moving Data Across the DB2 Family,″ an IBM Redbooks™ publication. This Redbooks publication can be found at the following URL: http://www.redbooks.ibm.com/redbooks/SG246905.

Chapter 9. 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 might 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 host or System i database server to the appropriate DB2 SQLCODEs.

If you want to turn off SQLCODE mapping, specify **NOMAP** in the parameter string of the DCS directory.

If you port an application directly from a host or System i database server, such as DB2 Universal Database for OS/390 and z/OS, you might want to turn off SQLCODE mapping. This would let you use the application without changing the SQLCODEs that it references.

Turning off SQLCODE mapping

If you want to turn off SQLCODE mapping, specify NOMAP in the parameter string of the DCS directory.

If you port an application directly from a host or System i database server, such as DB2 Universal Database (UDB) for OS/390 and z/OS, 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 host or System i database server to the appropriate DB2 SQLCODEs. The following files are copies of the default SQLCODE mapping:

- dcs1dsn.map maps DB2 Universal Database (UDB) for OS/390 and z/OS and DB2 for z/OS SQLCODEs.
- dcs1ari.map maps DB2 UDB for VSE and VM SQLCODEs.
- dcs1qsq.map maps DB2 UDB for iSeries and DB2 for i5/OS SQLCODEs.

No mapping is required for DB2 on Linux or UNIX operating systems.

1. If you want to override the default SQLCODE mapping or you are using a host or System i 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.
2. Specify the file name of your new SQLCODE mapping file in the parameter string of the DCS Directory.
3. 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]]
```

The *input_code* represents one of the following:

sqlcode The SQLCODE from the host or System i 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 System i 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. The *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 data type 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 data type 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 System i database server products support only CHAR tokens. The character **i** must be lowercase.

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

For example, the host or System i 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 9 on page 62 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 9. An SQLCODE Mapping File

The following descriptions correspond to the matching line number in the previous figure:

1. The SQLCODE is mapped from -007 to -007. The first input token received from the host or System i 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 System i 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 System i 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 System i 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 System i 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 10. Database system monitoring and DB2 Connect

Several ways to monitor connections and performance in an environment using DB2 Connect are discussed. The type of monitoring that is done is specific to the operating system.

Monitoring connections for remote clients

You can use the database system monitor with a DB2 Connect server product, such as 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 variable:

```
db2set DB2CONNECT_IN_APP_PROCESS=NO
```

For example, when an error occurs at the host or System i 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 unit of work id (UOWID), for two-phase connections protected by the DRDA-3 sync point 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.

Listing the Status of Monitor Switches

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

Monitoring performance using the Windows Performance Monitor

Windows operating systems provide 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 can save as many PMC files as you like, each reflecting a different cross-section of system performance.

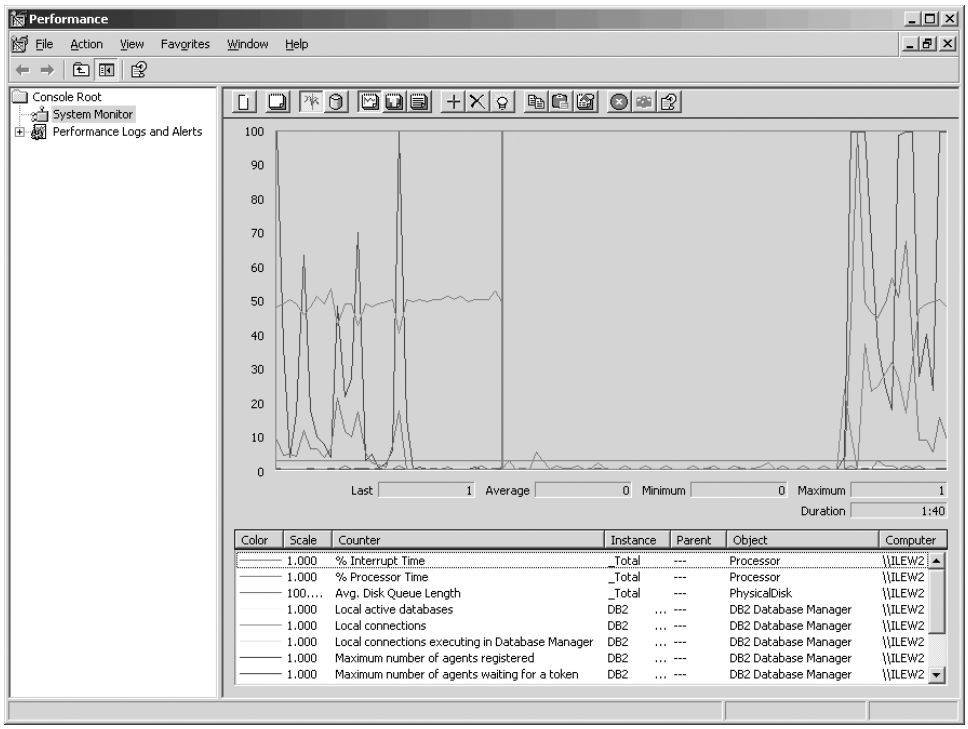


Figure 10. Performance Monitor

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

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 want 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        = 12-15-2001 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                     = 12-15-2001 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         = 12-15-2001 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        = 12-15-2001 10:28:25.592631
```

```
UOW start timestamp                         = 12-15-2001 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       = 12-15-2001 10:29:06.142790
Most recent operation stop timestamp        = 12-15-2001 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                   = 12-15-2001 10:29:06.142790
Statement stop timestamp                    = 12-15-2001 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)
```

DCS application status

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

In the output that follows, the format for the Host Application ID and Client Application ID can differ depending on the host or System i database version and the TCP/IP support level.

Table 11. Application ID format based on host version and TCP/IP support level

Scenario	Application ID format
Clients accessing data servers with RDB Manager Level support less than 7	`G91A0D3A.P8BC.060306212019`
Clients accessing data servers with RDB Manager level support 8 or greater over TCP/IP v4	`9.26.13.61.65289.060306213816`
Clients accessing data servers with RDB Manager level support 8 or greater over TCP/IP v6	`2002:91a:519:13:209:6bff:fe14:4fbb.7684.060306213741`

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 a TCP/IP connection (DB2 Connect to DB2 Universal Database for z/OS and OS/390):

```
Auth Id Application Name Appl.  Host Application Id
                         Handle
------- ---------------- ------ ------------------------------------------------
NEWTON  db2cli.exe       7      G91A0D3A.P8BC.060306212019
NEWTON  db2cli.exe       25     9.26.13.61.65289.060306213816
NEWTON  db2cli.exe       20     2002:91a:519:13:209:6bff:fe14:4fbb.7684.060306213741
```

Auth.Id
The authorization ID that was used to log on to the host or System i 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.

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. 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 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 System i 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
                                                     Handle
------------------------------ -------------------- ---------- ------------------------------------------------
NEWTON                          db2cli.exe           37         2002:91a:519:13:209:6bff:fe14:4fbb.8196.060306214224


Seq#  Client   Client   Client   Client     Host Application Id
      DB Alias Node     Release  Codepage
----- -------- -------- -------- ---------- --------------------------
00001 MDB      SAYYID   SQL09000 1252       G91A0D3A.P982.060306214231


Seq#  Host DB Name         Host
                           Release
----- -------------------- --------
00001 MEXICO               DSN08015
```

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.

This value lets you correlate connections from clients to the DB2 Connect workstation and from the DB2 Connect workstation to the host or System i 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 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:

- SQL07010 for Version 7.1 of DB2 Universal Database and DB2 Connect products and their clients.
- SQL08010 for Version 8.1 of DB2 Universal Database and DB2 Connect products and their clients.
- SQL08020 for Version 8.2 of DB2 Universal Database and DB2 Connect products and their clients.
- SQL09120 for Version 9.1 of DB2 products, 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 System i database server databases, the host or System i 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 System i database server product (for example, DSN for DB2 Universal Database for z/OS and OS/390, ARI for DB2 Server for VSE & VM, or QSQ for DB2 for i5/OS)

VV Represents a two-digit version number, such as `08`.

RR Represents a two-digit release number, such as `01`.

M Represents a one-character modification level (0-9 or A-Z).

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       = 2002:91a:519:13:209:6bff:fe14:4fbb.8196.060306214224
Sequence number                 = 00001
Authorization ID                = NEWTON
Trusted Authorization ID        =
Application name                = db2cli.exe
Application handle              = 37
Application status              = waiting for request
Status change time              = Not Collected
Client node                     = SAYYID
Client release level            = SQL09000
Client platform                 = NT
Client protocol                 = TCP/IP
Client codepage                 = 1252
Process ID of client application = 1192
Client login ID                 = ISAYYID
Host application ID             = G91A0D3A.P982.060306214231
Sequence number                 = 00001
Database alias at the gateway   = MDB
DCS database name               = MDB
Host database name              = MEXICO
Host release level              = DSN08015
Host CCSID                      = 1208
```

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 or System i 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 or System i 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 or System i 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.

Health monitor and alerts

DB2 for z/OS health monitor evaluates object maintenance policies periodically. If the health monitor determines the need for an object's maintenance, health alerts are created. Actions in response to health alerts can be viewed, submitted for running, and saved.

DB2 for z/OS health monitor overview

On z/OS systems, the DB2 for z/OS health monitor is started as a task for each DB2 subsystem to be monitored or on a dedicated member of a data sharing group.

The DB2 for z/OS health monitor triggers the evaluation of object maintenance policies at scheduled times and intervals, as defined in the policy. The object maintenance policies are created using the DB2 Control Center's Create Object Maintenance Policy wizard. During each policy evaluation, the criteria for recommending maintenance is checked against the thresholds set in the object maintenance policy to determine the need for object maintenance, that is, whether

COPY, REORG, RUNSTATS, STOSPACE, ALTER TABLESPACE, or ALTER INDEX are required, and to identify restricted states, such as CHKP, on table space, index, and storage group objects where applicable. When objects are identified to be in alert state during policy evaluation,the policy health alert contacts are notified at their e-mail addresses or pager numbers. The list of health alert contacts for each DB2 subsystem is defined in and managed from the Control Center.

A snapshot of the evaluation schedule for the policies, which is used by the health monitor to determine when to trigger policy evaluations, is initially taken by the health monitor when it is started. This schedule snapshot is refreshed at the refresh time specified when the health monitor was started, or when the health monitor receives a refresh command. Any change to the evaluation schedule of a policy is picked up by the health monitor when the schedule refresh occurs.

The health monitor is started and stopped from the console, using the MVS system START and STOP commands, respectively.

A sample cataloged procedure (DSNHMONP) that starts a DB2 health monitor, and a sample cataloged procedure (DSNHMONA) that starts multiple DB2 health monitors within an MVS system or Parallel Sysplex®, are placed in a procedure library by the installation job DSNTIJHM.

Views, tables, data sets, cataloged procedures, stored procedures, user-defined functions, and the result set table, which are used by the db2 health monitor or the related tasks listed below, are created and installed by the installation jobs DSNTIJCC and DSNTIJHM. DSNTIJCC and DSNTIJHM are shipped with FMIDs JDB771D and JDB881D.

Policy Evaluation Log

Policy evaluations triggered by the DB2 health monitor are logged in the table DSNACC.HM_EVAL_LOG. An entry is logged when a policy evaluation starts and when a policy evaluation ends. Log entries are kept for 7 days, after which they will be deleted from the table. The DB2 view DSNACC.HM_ALERT_PO_EV, which was created on this table by the DSNTIJCC installation job, can be used to display all policies whose last evaluation iteration was not successful.

Starting, stopping and refreshing the DB2 for z/OS health monitor

On the z/OS system, the DB2 for z/OS health monitor is started as a task for each DB2 subsystem to be monitored or on a dedicated member of a data sharing group.

- To start a DB2 health monitor, issue the following START MVS system command:

  ```
  S membername,DB2SSN=ssid,JOBNAME=HMONssid,TRACE=trace,REFRESH=nn
  ```

  TRACE and REFRESH parameters are optional.

  membername

  Specifies a procedure library member that is executed to start the DB2 health monitor, that is, DSNHMONP. This cataloged procedure is created by the DSNTIJHM installation job.

  ssid

  Specifies the name or identifier of the DB2 subsystem to be monitored.

trace

Specifies the trace flag. Possible values are:

– ON - Turn on trace. Trace records are written to SYSOUT

– OFF - Do not turn on trace

The default is OFF.

nn

Specifies the hour (using a 24-hour clock) when the health monitor refreshes the evaluation schedule snapshot it uses to trigger policy evaluations. The default is 22.

- To start multiple DB2 health monitors, issue the following START MVS system command:

  ```
  S membername
  ```

  membername

  A procedure library member that is executed to start multiple DB2 health monitors, that is, DSNHMONA.

  Note: Before starting multiple DB2 health monitors with one START command using DSNHMONA, the HMONPARM data set specified in the DSNHMONA proc must be populated with the list of subsystems to be monitored. The cataloged procedure and the data set are created by the DSNTIJHM installation job.

- To refresh the policy evaluation schedule snapshot used by the DB2 health monitor to determine when to trigger policy evaluations, issue the following MODIFY MVS system command:

  ```
  F HMONssid,APPL=REFRESH
  ```

  ssid

  Name or identifier of the DB2 subsystem that the DB2 health monitor you're refreshing is monitoring.

- To stop a DB2 health monitor, issue the following STOP MVS system command:

  ```
  STOP HMONssid  or P HMONssid
  ```

  ssid

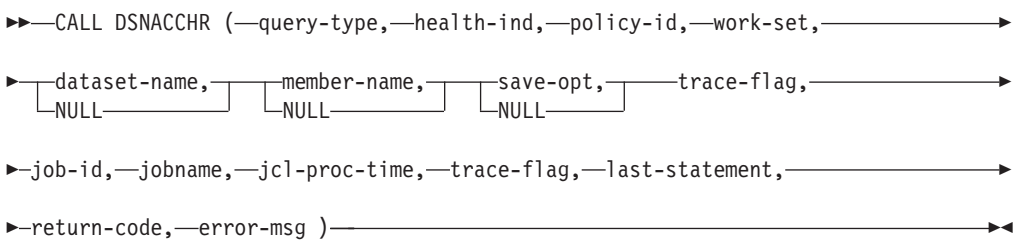
  Name or identifier of the DB2 subsystem that the DB2 health monitor you're stopping is monitoring.

Viewing, submitting, and saving recommended actions

To view, submit, and save the actions recommended for alert objects identified during policy evaluation, call the DB2 stored procedure SYSPROC.DSNACCHR, which is created by the DSNTIJCC installation job. DSNACCHR is a stored procedure which determines the recommended actions for alert objects identified during policy evaluation and generates a JCL job that will execute the recommended actions.

The following syntax diagram shows the SQL CALL statement for invoking DSNACCHR. Because the linkage convention for DSNACCHR is GENERAL WITH NULLS, if you pass parameters in host variables, you need to include a null indicator with every host variable. Null indicators for input host variables must be initialized before you execute the CALL statement.

Syntax



query-type

Specifies what you want to do with the actions recommended for objects identified to be in alert state during policy evaluation. Possible values are:

- 0 - View recommended actions on alert objects as a JCL job
- 1 - Submit the JCL job that executes the recommended actions on alert objects
- 2 - Submit the JCL job that executes the recommended actions on alert objects, and put the job on the hold queue
- 3 Save recommended actions on alert objects as a JCL job in a library member

query-type is an input parameter of type INTEGER.

health-ind

Specifies the type of alert that DSNACCHR includes in the JCL job. Possible values are:

- RS - Restricted State
- EX - Extents Exceeded
- RR - REORG Required
- CR - COPY Required
- RT - RUNSTATS Required
- SS - STOSPACE Required

health-ind is an input parameter of type VARCHAR(4).

policy-id

Specifies an object maintenance policy. *policy-id* is an input parameter of type VARCHAR(7).

work-set

Specifies the work set of an object maintenance policy that identified the alert objects that DSNACCHR includes in the JCL job. This work set must be identified with the policy and type of alert specified in the parameters *policy-id* and *health-ind*. *work-set* is an input parameter of type INTEGER.

dataset-name

Specifies a fully qualified partitioned data set (PDS) or partitioned data set extended (PDSE) name. This value must be specified if *query-type* is 3. *dataset-name* is an input parameter of type VARCHAR(44).

member-name

Specifies a member of the partitioned data set (PDS) or partitioned data set extended (PDSE) specified in the *dataset-name* parameter where the object maintenance JCL job will be saved.This value must be specified if *query-type* is 3. *member-name* is an input parameter of type VARCHAR(8).

save-opt

Specifies how to save the object maintenance JCL job. This value must be specified if *query-type* is 3. Possible values are:

- R - Replace
- A - Append
- NM - New member

save-opt is an input parameter of type VARCHAR(2).

trace-flag

Specifies whether tracing will be turned on or off. Possible values are:

- Y - Turn trace on
- N - Turn trace off

trace-flag is an input parameter of type CHAR(1).

job-ID

When *query-type* is 1 or 2, specifies the job ID of the submitted job. *job-id* is an output parameter of type VARCHAR(8).

jobname

When *query-type* is 1 or 2, specifies the name of the submitted job. *jobname* is an output parameter of type VARCHAR(8).

jcl-proc-time

Specifies the time request was processed. *jcl-proc-time* is an output parameter of type TIMESTAMP.

last-statement

When DSNACCHR returns a severe error (return code 12), this field contains the SQL statement that was executing when the error occurred. *last-statement* is an output parameter of type VARCHAR(2500).

return-code

The return code from DSNACCHR execution. Possible values are:

- 0 - DSNACCHR executed successfully
- 12 - DSNACCHR terminated with severe errors. The *error-msg* parameter contains a message that describes the error. The *last-statement* parameter contains the SQL statement that was executing when the error occurred.

return-code is an output parameter of type INTEGER.

error-msg

When DSNACCHR returns a severe error (return code 12), this field contains error messages, including the formatted SQLCA. *error-msg* is an output parameter of type VARCHAR(1331).

DSNACCHR returns one result set when the *query-type* parameter is 0. The result set contains the JCL job generated by DSNACCHR. The DSNACCHR result set table is created by the DSNTIJCC installation job. Table 12 shows the format of the result set.

Table 12. DSNACCHR result set format

Column name	Data type	Description
JCLSEQNO	INTEGER	Sequence number of the table row (1,...,n)
JCLSTMT	VARCHAR(80)	Specifies a JCL statement

Viewing health alert summaries

The HEALTH_OVERVIEW function returns information from the Health Alert Summary VSAM KSDS data set as a DB2 table. This data set is created by the DSNTIJHM installation job.

The Health Alert Summary data set contains information about the state of the DB2 health monitor and alert summary statistics for every DB2 subsystem previously or currently monitored by the health monitor on that MVS system or Parallel Sysplex. These information are returned to the client with a row for each DB2 subsystem and alert recommendation.

The result of the function is a DB2 table with the following columns:

ip-addr

The IP address of the DB2 server. This is a column of type VARCHAR(40).

db2-ssid

The subsystem identifier of the DB2 subsystem. This is a column of type VARCHAR(4).

health-ind

The type of alert. Possible values are:

- RS - Restricted State
- EX - Extents Exceeded
- RR - REORG Required
- CR - COPY Required
- RT - RUNSTATS Required
- SS - STOSPACE Required
- PO - Failed Policy Evaluation
- HM - Health Monitor State

health-ind is a column of type VARCHAR(4).

host-name

The fully qualified domain name of the DB2 server. This is a column of type VARCHAR(255).

summary-stats

The state of the DB2 health monitor if *health-ind* is 'HM'. Possible values are:

- 0 Health monitor is not started
- 1 Health monitor is started
- -1 Health monitor state is unknown

Otherwise, the total number of alert objects with the alert type specified in *health-ind*. This is a column of type INTEGER.

alert-state

The state of the alert specified in *health-ind*. Possible values are:

- 5 - Alarm
- 4 - Attention
- 3 - Warning
- 0 - Normal

alert-state is always 0 when *health-ind* is 'HM'. This is a column of type INTEGER.

The external program name for the function is HEALTH_OVERVIEW, and the specific name is DSNACC.DSNACCHO. This function is created by the DSNTIJCC installation job.

Example: Find the total number of alert objects requiring COPY for the DB2 subsystem 'ABCD':

```
SELECT SUMMARYSTATS FROM TABLE (DSNACC.HEALTH_OVERVIEW()) AS T
      WHERE DB2SSID = 'ABCD'
              AND HEALTHIND = 'CR';
```

Viewing health alert objects

Alert objects identified during the last successful iteration of a policy evaluation are saved in these alert object repository tables, depending on their object type.

The alert objects are:

- DSNACC.HM_MAINT_TS for table spaces
- DSNACC.HM_MAINT_IX for indexes
- DSNACC.HM_MAINT_SG for storage groups

DB2 creates a number of views on these alert object repository tables. The views and alert object repository tables are created by the DSNTIJCC installation job. Table 13 lists the tables on which each view is defined and the view descriptions. All view names and table names have the qualifier DSNACC.

Table 13. Views on health alert objects

View Name	On Table	View Description
HM_ALERT_TS_RS	HM_MAINT_TS	Displays all table spaces in restricted state
HM_ALERT_TS_EX	HM_MAINT_TS	Displays all table spaces whose extents have exceeded a user-specified limit

Table 13. Views on health alert objects (continued)

View Name	On Table	View Description
HM_ALERT_TS_RR	HM_MAINT_TS	Displays all table spaces that require REORG
HM_ALERT_TS_CR	HM_MAINT_TS	Displays all table spaces that require COPY
HM_ALERT_TS_RT	HM_MAINT_TS	Displays all table spaces that require RUNSTATS
HM_ALERT_IX_RS	HM_MAINT_IX	Displays all indexes that are in restricted state
HM_ALERT_IX_EX	HM_MAINT_IX	Displays all indexes whose extents have exceeded a user-specified limit
HM_ALERT_IX_RR	HM_MAINT_IX	Displays all indexes spaces that require REORG
HM_ALERT_IX_CR	HM_MAINT_IX	Displays all indexes that require COPY
HM_ALERT_IX_RT	HM_MAINT_IX	Displays all indexes that require RUNSTATS
HM_ALERT_SG_SS	HM_MAINT_SG	Displays all storage groups that require STOSPACE

Part 3. High availability and DB2 Connect

There are specific considerations with respect to high availability in an environment that uses DB2 Connect. If, for some reason, a database server in a network becomes unavailable for use then the ability to reroute a client workstation to an alternate database server in the network is important.

Chapter 11. High availability and load balancing for host database connectivity

In today's information technology market, there is a high demand for around the clock availability of data. This demand must be met in order for a business to compete with its competitors and maintain continued growth. Many of today's web, e-business, and spreadsheet applications require access to enterprise data. A reliable, fast and secure connection to host and System i databases must be established. This connection must be constantly available and be able to handle the high connection demands under critical load conditions. How can this connection be built?

High availability scenario

A company has several workstations and application servers running on Windows, Linux, and UNIX. These machines require access to data residing on several host and System i databases. Applications running on these machines demand fast and reliable connections to the databases. The entire system is connected by an Ethernet network using TCP/IP.

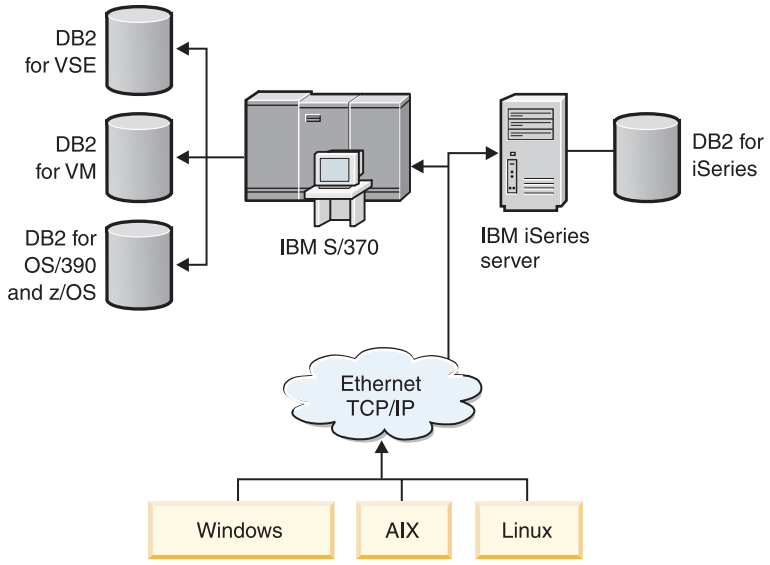


Figure 11. Sample network scenario

For workstations and application servers to access host and System i databases, you need a connectivity component as an intermediary. This component must provide a highly available, robust, and fast connection to host and System i databases. It must also be scalable to anticipate for future growth in connection volume.

Use the related links from this topic to see details regarding a solution using DB2 Connect and the automatic client reroute feature .

Chapter 12. Automatic client reroute description and setup (DB2 Connect)

The main goal of the automatic client reroute feature is to enable an IBM Data Server Client application to recover from a loss of communications so that the application can continue its work with minimal interruption. As the name suggests, rerouting is central to the support of continuous operations. But rerouting is only possible when there is an alternate location that is identified to the client connection. In a non-DB2 Connect high-availability environment, the database being accessed is typically synchronized between the original DB2 server and the alternate DB2 server through one of various means, such as High availability cluster multiprocessor (HACMP™) or High availability disaster recovery (HADR).

However, in the case of the DB2 Connect server, because there is no requirement for the synchronization of local databases, you only need to ensure that both the original and alternate DB2 Connect servers have the target host or System i database catalogued in such a way that it is accessible using an identical database alias.

Note: In a DB2 Connect server environment, an alternate DB2 Connect server can be specified to enable automatic rerouting between a client and the DB2 Connect server. For rerouting to occur between the DB2 Connect personal or server products and a host or System i database server, the remote server must provide one or more alternate addresses for itself. In the case of DB2 for z/OS, multiple addresses are known if the database is a Sysplex data sharing environment.

Rerouting capability for Sysplex can be configured between DB2 Connect and the host database server if Sysplex support is enabled. The rerouting capability for Sysplex is a DB2 Connect feature that allows DB2 Connect to retry the connection against other members of the Sysplex group following the loss of communication with the original member. The alternate server does not need to be cataloged in the database directory to enable the reroute capability for Sysplex on DB2 Connect. By default, the reroute capability for Sysplex is enabled if Sysplex support is enabled.

In order for an IBM Data Server Client to have the ability to recover from a loss of communications to a DB2 Connect server using automatic client reroute, an alternate DB2 Connect server location must be specified before the loss of communication occurs. The UPDATE ALTERNATE SERVER FOR DATABASE command is used to define the alternate DB2 Connect server location for a particular host or System i database. The alternate hostname and port number is given as part of the command. The location is stored in the system database directory file at the DB2 Connect server. In order to ensure the alternate DB2 Connect server location specified applies to that database for all clients, the alternate server location has to be specified at the DB2 Connect server side. The alternate server is ignored if it is set at the client instance.

For example, suppose a host or System i database is catalogued using a database alias of `db1` at a DB2 Connect server `S1` (with a hostname of `db2conn1` and a port number of 122). The database administrator would like to specify an alternate DB2 Connect server `S2` at hostname `db2conn2` with a port number of 123. Here is the command the database administrator would run at the DB2 Connect server `S1`:

```
db2 update alternate server for database db1 using hostname db2conn2 port 123
```

After you have specified the alternate DB2 Connect server location for database alias db1 at DB2 Connect server S1, the alternate server location information is returned to the IBM Data Server Client as part of the connection process. If communication between the IBM Data Server Client and the DB2 Connect server S1 is lost for any reason (typically a communication error, such as SQL code -30081 or SQL code -1224), the IBM Data Server Client will attempt to reconnect to db1 through either the original DB2 Connect server (S1) or the alternate DB2 Connect server (S2), alternating the attempts between the two servers. The time interval between attempts starts off rapidly, then gradually lengthens with each attempt.

Once a connection is successful, the SQL code -30108 is returned to indicate that a database connection has been reestablished following the communication failure. The hostname or IP address and service name or port number are returned. The IBM Data Server Client only returns the error for the original communications failure to the application if the reestablishment of the client communications is not possible to either the original or alternative server.

The following considerations involving alternate server connectivity in a DB2 Connect server environment should also be noted:

- When using a DB2 Connect server for providing access to a host or System i database on behalf of both remote and local clients, confusion can arise regarding alternate server connectivity information in a system database directory entry. To minimize this confusion, consider cataloging two entries in the system database directory to represent the same host or System i database. Catalog one entry for remote clients and catalog another for local clients.
- Any SYSPLEX information that is returned from a target DB2 for z/OS server is kept only in cache at the DB2 Connect server. Only one alternate server is written to disk. When multiple alternates or multiple active servers exist, the information is only maintained in memory and is lost when the process terminates.

Chapter 13. Configuring automatic client reroute for client connection distributor technology

Distributor or dispatcher technologies such as WebSphere EdgeServer distribute client application reconnection requests to a defined set of systems if a primary database server fails. If you are using distributor technology with DB2 automatic client reroute, you must identify the distributor itself as the alternate server to DB2 automatic client reroute.

You might be using distributor technology in an environment similar to the following:

Client —> distributor technology —> (DB2 Connect Server 1 or DB2 Connect Server 2) —> DB2 z/OS

where:

- The distributor technology component has a TCP/IP host name of DThostname
- The DB2 Connect Server 1 has a TCP/IP host name of GWYhostname1
- The DB2 Connect Server 2 has a TCP/IP host name of GWYhostname2
- The DB2 z/OS server has a TCP/IP host name of zOShostname

The client is catalogued using **DThostname** in order to utilize the distributor technology to access either of the DB2 Connect Servers. The intervening distributor technology makes the decision to use **GWYhostname1** or **GWYhostname2**. Once the decision is made, the client has a direct socket connection to one of these two DB2 Connect gateways. Once the socket connectivity is established to the chosen DB2 Connect server, you have a typical client to DB2 Connect server to DB2 z/OS connectivity.

For example, assume the distributor chooses **GWYhostname2**. This produces the following environment:

Client —> DB2 Connect Server 2 —> DB2 z/OS

The distributor does not retry any of the connections if there is any communication failure. If you want to enable the automatic client reroute feature for a database in such an environment, the alternative server for the associated database or databases in the DB2 Connect server (DB2 Connect Server 1 or DB2 Connect Server 2) should be set up to be the distributor (DThostname). Then, if DB2 Connect Server 1 locks up for any reason, automatic client rerouting is triggered and a client connection is retried with the distributor as both the primary and the alternate server. This option allows you to combine and maintain the distributor capabilities with the DB2 automatic client reroute feature. Setting the alternate server to a host other than the distributor host name still provides the clients with the automatic client reroute feature. However, the clients will establish direct connections to the defined alternate server and bypass the distributor technology, which eliminates the distributor and the value that it brings.

The automatic client reroute feature intercepts the following SQL codes:

- sqlcode -20157
- sqlcode -1768 (reason code = 7)

Note: Client reroute might not be informed of socket failures in a timely fashion if the setting of the ″TCP Keepalive″ operating system configurations parameter is too high. (Note that the name of this configuration parameter varies by platform.)

Part 4. Tuning and DB2 Connect

A database environment that uses DB2 Connect to move database requests and responses between client workstations and database servers has specialized concerns when considering performance issues. There are several ways to improve or maintain performance in this environment.

Chapter 14. DB2 Connect performance considerations

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 System i 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, is an example of a software resource.

Data Flows

Figure 12 shows the path of data flowing between the host or System i database server and the workstation through DB2 Connect.

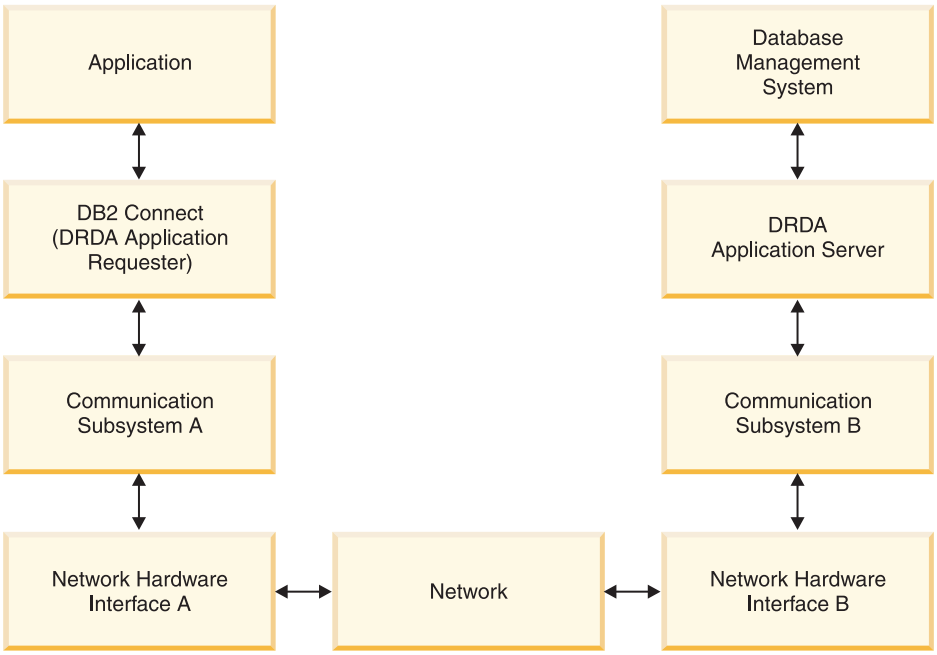


Figure 12. Data Flows in DB2 Connect

- The host or System i 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 might 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 System i server. 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.
- DB2 Connect and the communication subsystem A are usually located on the same system. For the scope of this discussion, it is assumed 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.

Benchmarking

Benchmarking compares 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 than those being measured unless the scenario includes some 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 tables list 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 14. Performance tools for CPU and memory usage

System	Tool	Description
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	
Windows	Microsoft Performance Monitor	

Table 15. Performance tools for database activity

System	Tool	Description
All	Database monitor	Determines if the problem originates from the database.
OS/390 or zSeries	DB2PM (IBM), OMEGAMON/DB2 (Candle®), TMON (Landmark), INSIGHT (Goal Systems) and DB2AM (BMC)	
Windows	Microsoft Performance Monitor	

Table 16. Performance tools for network activity

System	Tool	Description
AIX	netpmon	Reports low level network statistics, including TCP/IP statistics such as the number of packet or frames received per second.
Network controller such as 3745	NetView® Performance Monitor	Reports utilization of communication control and VTAM®.
Linux and UNIX	netstat	Handles TCP/IP traffic.

Chapter 15. Optimizing ODBC access

DB2 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 Configuration Assistant (CA).

To activate the optimized ODBC:

- If you are defining a new connection:
  1. Start the DB2 CA.
  2. Open the Selected menu and select Add Database Using Wizard...
  3. Follow the wizard's pages until you get to the **Data Source** page.
  4. Check **Register this database for CLI/ODBC**.
  5. Specify how CLI/ODBC applications accessing this database should be registered:
     - **As system data source** means the database is available to all users on the system.
     - **As user data source** means you are the only user who can access the database.
     - **As file data source** means a file containing data source information will be created. This data source file can be shared with other workstations if you have a TCP/IP connection. Otherwise, the file can only be used on this computer
  6. Type a **Data source name**.
  7. (Optionally) Select an application from the **Optimize for application** list to optimize the data source settings for a particular application.
  8. Click **OK** and exit the CA.
- If you are updating an existing connection:
  1. Start the DB2 CA.
  2. Double-click the database alias that you want to optimize.
  3. Click **Data Source**.
  4. Check **Register this database for CLI/ODBC**.
  5. Specify how CLI/ODBC applications accessing this database should be registered:
     - **As system data source** means the database is available to all users on the system.
     - **As user data source** means you are the only user who can access the database.
     - **As file data source** means a file containing data source information will be created. This data source file can be shared with other workstations if you have a TCP/IP connection. Otherwise, the file can only be used on this computer
  6. Type a **Data source name**.
  7. (Optionally) Select an application from the **Optimize for application** list to optimize the data source settings for a particular application.
  8. Click **OK** and exit the CA.

Chapter 16. Application design

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

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
```

Stored procedures help to reduce network traffic by placing program logic at the server. You can commit automatically when exiting the procedure. You can also return results sets, which minimize application logic at the client.

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 System i database server and the DB2 Connect server. 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.

Within DB2 Connect, you can control the amount of data that is transferred within each block.

To invoke blocking, use the BLOCKING option of the prep or bind command. Blocking is on, if:

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

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

SQL statements with BLOCKING

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

An updatable 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 might 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.

Other SQL considerations

Using the Command Line Processor (CLP) is, in general, slower than having dynamic SQL in the program because the CLP must parse the input before submitting the SQL to the database engine. The CLP also formats data when it is received, which might 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 might improve performance for applications that access more than one database.

Chapter 17. Connection management

Connection management is composed of two operations: connection pooling and connection concentrator. Connection pooling reduces the overhead of database connections and manages connection volume. Connection concentrator increases the scalability within your working environment by optimizing the resources used by the host database servers. Both operations are discussed here.

Connection pooling

DB2 Connect server products, such as DB2 Connect Enterprise Edition, 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 problem 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 server products use connection pooling to maintain open connections to the database in a readily accessible pool.

Most applications based on web technologies execute large volume of short transactions. A typical web transaction is executed as part of its own connection. In other words, executing a transaction means establishing a database connection and then terminating this connection only after a few SQL statements. This process of establishing and tearing down a connection is very costly. It involves creation of a DB2 Connect agent, establishing a network connection between this agent and the DB2 server, and creation of a DB2 thread on the server. For longer running connections these costs are amortized over all of the transactions executed on this connection but for a typical web transaction these costs will typically exceed the cost of executing the transaction itself.

Connection pooling is a technique that allows reuse of an established connection infrastructure for subsequent connections. When a DB2 Connect instance is started a pool of coordinating agents is created. When a connection request comes in an agent is assigned to this request. The agent will connect to the DB2 server and a thread will be created in DB2. When the application issues a disconnect request, the agent will not pass this request along to the DB2 server. Instead, the agent is put back in to the pool. The agent in the pool still owns its connection to the DB2 server and the corresponding DB2 thread. When another application issues a connect request, this agent is assigned to this new application. To insure secure operation, user identity information is passed along to the DB2 thread which, in turn, performs user authentication.

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.

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.

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 want the system to maintain. Setting this parameter to zero is equivalent to turning off the connection pooling feature. The default for this configuration parameter is set to AUTOMATIC with a value of 100. By being set to AUTOMATIC, DB2 Connect automatically manages the number of idle agents in the idle agent pool.

DB2 Connect does not establish connections to the database before receiving its first client request. Alternatively, you can fill the pool of idle agents before any clients make a request. The pool can be filled on startup using the **num_initagents** configuration parameter. This parameter determines how many idle agents should be created at start up time. These idle agents initially will not 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 default for this configuration parameter is set to AUTOMATIC with a value of 200. By being set to AUTOMATIC, DB2 Connect automatically manages the number of coordinator agents.

The DB2 registry variable DB2CONNECT_IN_APP_PROCESS allows applications running on the same machine as a DB2 Connect server product to either have DB2 Connect run within the applications process, default behavior, or to have the application connect to the DB2 Connect server product and then have the host connection run within an agent. For an application to use connection pooling the connections to the host must be made from within the DB2 Connect server product agents and thus DB2CONNECT_IN_APP_PROCESS must be set to NO.

DB2 Connect Connection Pooling versus Application Server Connection Pooling

Connection pooling is a must for any web technologies based application that is to support large volumes of transactions. Most web application servers now provide their own way of pooling database connections. For example, both Microsoft MTS (COM+) and IBM WebSphere provide connection pooling.

Application pooling mechanisms implemented by these servers differ significantly from what is provided by the DB2 Connect servers. Since application servers pool

connections only for their own use they typically presume that user id, password, isolation levels, and so on, will be exactly the same for all connections. Even more important, application servers only pool connections initiated by the same process. This means that connections from other machines, users or processes are not pooled. While these application server pooling techniques are effective for reusing connections established by the same instance of an application they are absolutely ineffective for pooling connections from multiple users, servers, and so on.

Connection pooling, provided by the DB2 Connect servers, is completely application, machine and user independent. Connections from multiple clients, application servers all with different user IDs can all reuse each other's connections resulting in a much better utilization of the pooled resources.

Which type of connection pooling is the right one to use? Both. Generally, using both DB2 Connect connection pooling and Application Server connection pooling is a good strategy since they don't interfere with each other. Even when application server connection pooling is enabled, DB2 Connect connection pooling can provide connection reuse for multiple application servers as well as other clients using the DB2 Connect server.

Connection concentrator

The connection concentrator reduces the resources required on DB2 for OS/390 and z/OS database servers to support large numbers of workstation and web users. This function can dramatically increase the scalability of your DB2 for OS/390 and z/OS and DB2 Connect solution while also providing for fail-safe operation and transaction level load balancing in DB2 for OS/390 and z/OS data sharing environments.

The connection concentrator allows applications to stay connected without any resources being consumed on the DB2 host server. You can have thousands of users active in applications and only have a few threads active on the DB2 host server.

DB2 Connect's *connection concentrator* technology allows DB2 Connect server products, such as DB2 Connect Enterprise Edition, to provide support to thousands of users simultaneously executing business transactions, while drastically reducing resources required on the S/390 host or System i database servers. It accomplishes this goal by concentrating the workload from all applications in a much smaller number of S/390 host or System i database server connections. While this might 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 concentrator takes the concept of an agent and splits it into two entities:

- Logical agent, which represents an application connection.
- Coordinating agent, which owns the DB2 connection and thread, and executes application requests.

When a new application attempts a connection to the host, it is assigned a logical agent. To pass SQL to the database, a coordinating agent is required and is assigned as soon as a new transaction is initiated. The key to this architecture is the fact that the coordinating agent is:

- Disassociated from the logical agent

- Returned to the pool when transaction completes due to a commit or rollback

Another key feature is the method of assigning coordinating agents to new transactions in a data sharing environment. DB2 Connect implements a sophisticated scheduling algorithm that uses OS/390 and z/OS Work Load Manager (WLM) information. This information is used to distribute workload across members of a data sharing group according to criteria set up in WLM. WLM is not only aware of the load on each member but also their availability. This allows DB2 Connect to transparently relocate work away from failed or overloaded members to members that are up and underutilized. DB2 Connect connection concentrator is activated when you set the number of maximum logical agents (*max_connections*) higher than the number of coordinating agents (*max_coordagents*).

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.

Alternatively the connection concentrator 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.

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 increases, 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 the transaction boundary end the association and return to the available pool.

An entity known as the *dispatcher* assigns worker agents to logical agents. Limitations in the number of open file handles on certain computing platforms might result in more than one scheduler instance.

Restrictions for the connection concentrator

There are a number of important restrictions to the use of the DB2 Connect server concentrator. Review the following information in its entirety before attempting to use the connection concentrator on your system.

General restrictions:

- 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 named pipes, but you will not be able to use its XA concentration features with that connection.
- For XA tightly coupled transaction support, all applications that participate in the same XA transaction must use the same DB2 Connect Server Instance to connect to the host.
- Only applications that close withhold resources (such as withhold cursors) on transaction boundaries 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.
- If you declare global temporary tables, they must be closed explicitly at transaction or branch boundary. Failure to close the tables will turn off connection concentration but the application will continue to work.
- All applications participating in the same XA transaction must have the same CCSID and use the same user ID 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.
- The concentrator only supports dynamic SQL from the Call Level Interface (CLI). CLI applications should also not use KEEPDYNAMIC as the concentrator depends on statements being re-prepared on each transaction boundary.
- 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.

When working with DB2 Version 9 or Version 8 FixPak 13 (or higher), to enable DB2 Connect concentrator support requires System i Version 5 Release 4 (PTF SI23726). Otherwise, only the XA portion of the connection concentrator is supported.

Activating the connection concentrator

The database manager configuration parameter *max_coordagents* sets the maximum number of logical agents. You can activate the concentrator feature by setting the value of *max_connections* to any number greater than the default. The default value for *max_connections* is equivalent to the value of *max_coordagents*. Because each application will have one logical agent, *max_connections* 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_connections* will take a numeric range from *max_coordagents* up to 64,000. The default number of logical agents is equal to *max_coordagents*.

Both *max_connections* and *max_coordagents* can be set to AUTOMATIC. If *max_connections* is set to AUTOMATIC, the number of connections can be increased beyond the base configured value. If both *max_connections* and *max_coordagents*are set to AUTOMATIC, *max_connections* can be increased beyond the base value, and *max_coordagents* is automatically increased to maintain the concentration ratio between connections and the coordinator agents.

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

max_coordagents
Maximum number of active coordinator agents.

num_poolagents
Agents pool size. The agent pool includes inactive agents and idle agents. For improved performance, *num_poolagents* should be configured to equal the average number of clients.

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 z/OS and DB2 for System i. 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 of XA transaction support

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 might 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_CONNECTIONS =  4,000
   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 might 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, assume that a TP Monitor is being used with a DB2 Connect gateway and an zSeries or System i 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. 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 might 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 can 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.

Connection pooling and connection concentrator

While connection pooling and connection concentrator seem to have similarities, they differ in their implementation and address different issues. Connection pooling helps reduce the overhead of database connections and handle connection volume. Connection concentrator helps increase the scalability of your DB2 for OS/390 and z/OS and DB2 Connect solution by optimizing the use of your host database servers.

When using connection pooling, the connection is only available for reuse after the application owning the connection issues a disconnect request. In many 2-tier client-server applications users do not disconnect for the duration of the workday. Likewise, most application servers in multi-tier applications establish database connections at server start up time and do not release these connections until the application server is shut down.

In these environments, connection pooling will have little, if any, benefit. However, in web and client-server environments where the frequency of connections and disconnections is higher than connection pooling will produce significant performance benefits. The connection concentrator allocates host database resources only for the duration of an SQL transaction while keeping user applications active. This allows for configurations where the number of DB2 threads and the resources they consume can be much smaller than if every application connection had its own thread.

When it comes to fail-safe operation and load balancing of workload connection concentrator is clearly the right choice as it allows reallocation of work with every new transaction. Alternatively, connection pooling can only offer very limited balancing and only at connect time.

Connection pooling and connection concentrator should be used together although they address different issues.

Connection concentrator required with WebSphere MQ Transaction Manager and DB2 for OS/390

When running applications in an IBM WebSphere MQ (formerly known as IBM MQSeries) environment, WebSphere MQ can act as an XA-compliant transaction manager, coordinating any distributed, two-phase commit transactions. When WebSphere MQ is acting as a transaction manager in this way, and the data sources are from the DB2 family of products, there are several configuration requirements.

Most of the configuration requirements in such a transaction manager environment are already documented elsewhere. For example, you must set the DB2 configuration parameter TP_MON_NAME to "MQ" at the DB2 runtime client.

However, there is a configuration requirement that was missing. The requirement is specific to DB2 Connect when connecting to data sources that are DB2 for OS/390 servers: when using WebSphere MQ to coordinate distributed transactions involving DB2 for z/OS and DB2 for i5/OS servers, the DB2 Connect connection concentrator feature must be enabled at the gateway. The connection concentrator is enabled when the value of the **max_connections** configuration parameter is greater than the value of the **max_coordagents** configuration parameter.

If you do not enable the connection concentrator, unexpected transaction behavior will result.

Chapter 18. DB2 Connect Sysplex support

A Sysplex is a collection of System z™ servers that cooperate, using hardware and software, to process work. The Sysplex coordinates the cooperation by increasing the number of processors working together, which increases the amount of work that can be processed. In addition to an increase in processing capability, a Sysplex can provide flexibility in mixing levels of hardware and software, and in dynamically adding systems.

Sysplex permits DB2 Connect to seamlessly balance connections across different members of a data sharing group. Sysplex also provides DB2 Connect the means to try alternate members should a failure occur with one member. The rerouting capability for Sysplex is a DB2 Connect feature. DB2 Connect support for Sysplex is enabled by default and so is the rerouting capability for Sysplex. Sysplex support to a host database can be turned off by removing the SYSPLEX parameter from its DCS directory entry, but the DCS entry itself should not be removed, even if it has no other parameter specified.

With the automatic client reroute capability for Sysplex, the default behavior is for a Sysplex enabled connection to retry on the connect when there is a communication failure. Special register values, up until the last successful transaction not holding resources, are replayed when DB2 Connect is connected to a DB2 for z/OS server.

You can configure the exact automatic client reroute retry behavior, including disablement, by using the DB2_MAX_CLIENT_CONNRETRIES and DB2_CONNRETRIES_INTERVAL registry variables. The connection timeout registry variable is DB2TCP_CLIENT_CONTIMEOUT.

Considerations for OS/390 and zSeries SYSPLEX exploitation

DB2 Connect provides load balancing and fault-tolerance when routing connections to multiple Sysplexes. When connected to a DB2 for OS/390 and z/OS database server running in a data sharing environment, DB2 Connect will spread the workload amongst the different DB2 subsystems comprising the data sharing group, based on the system load information provided by the Workload Manager (WLM).

DB2 Connect receives a prioritized list of Sysplex members from the WLM. Each Sysplex returns weighted priority information for each connection address. This list is then used by DB2 Connect to handle the incoming CONNECT requests by distributing them among the Sysplex members with the highest assigned priorities. For load balancing, the list of Sysplex weighted priority information is obtained during each connection. If the DB2 Connect connection concentrator is enabled, this list is also used when determining where to send each transaction.

Note: OS/390 and z/OS Distributed Data Facility (DDF) configuration does not need to be changed to take advantage of the DB2 Connect Sysplex exploitation.

DB2 Connect also provides fault-tolerance by attempting to connect to an alternate sysplex machine in the event of a connection failure. An error will only be returned to the application if all known connections have failed.

DB2 Connect Sysplex is designed with agent pooling in mind. With Sysplex enabled, DB2 Connect routes connections to another DDF member in the event that the connection to a participating member is lost. The reroute is accomplished according to a Sysplex server list.

With the addition of the concentrator, DB2 Connect now has the ability to balance the workload at transaction boundaries. The DB2 Connect concentrator must be enabled for this to work.

DB2 Sysplex exploitation

In a typical scenario, a DB2 Connect server (server A) would be in conversation with a Sysplex containing two DB2 for OS/390 and z/OS servers (servers B and C).

Sysplex server B	Sysplex server C
HOST_NAME=MVSHOST	HOST_NAME=MVSHOST1

Suppose that in this scenario an application now issues:

```
db2 connect to aliasb user xxxxxxx using xxxxxxxx
```

The connection to database `MVSHOST` is established. Because Sysplex exploitation is enabled both for the DB2 Connect server and the DCS directory entry, DB2 for OS/390 and z/OS identifies the network addresses to DB2 Connect for each Sysplex participant (MVSHOST and MVSHOST1. DRDA4 protocols and message flows are used to return this information). Once an initial connection has been made, the returned list of addresses is cached at the DB2 Connect workstation. Once the initial `CONNECT` is issued for a TCP/IP node, then the IP addresses are returned.

Priority information used for load balancing and fault tolerance

The list of addresses provided by DB2 for OS/390 and z/OS also includes priority information, including the number of connections for each network address. The list is refreshed whenever a new connection is made by DB2 Connect. This additional information is used for load balancing purposes, as well as for fault tolerance.

Cached Address List used by DB2 Connect

If the database connection to ALIASB fails, then an error message SQL30081N is issued, and the connection will be dropped. If a further connection request is received for ALIASB, DB2 Connect does the following:

1. It tries the highest priority server from the cached list of addresses based on the priority information that was returned by DB2 for OS/390 and z/OS. This strategy is always used by DB2 Connect, and it is by this means that load balancing is achieved.
2. If this connection attempt fails, then the other addresses in the list are tried, in descending order of priority, as returned by DB2 for OS/390 and z/OS. This is how DB2 Connect exploits the Sysplex information to achieve fault tolerance.
3. If all other attempts to connect fail, then DB2 Connect will retry the connection to ALIASB using the address contained in the cataloged node directory.

The db2pd command with the `sysplex` parameter (db2pd -sysplex) can be used for retrieving information about servers associated with a Sysplex environment.

Configuration requirements for Sysplex

Sysplex exploitation will not be used for a given database unless the DCS directory entry for that database contains Sysplex (not case-sensitive) in the 6th positional parameter.

Chapter 19. DB2 Connect tuning

Various parameters in the database manager configuration file can be used to tune DB2 Connect.

RQRIOBLK

The **RQRIOBLK** parameter sets the maximum size of network I/O blocks. A larger block size might 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 server. This increases the size of the working set and might 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 monitoring tools (such as the vmstat tool for Linux and UNIX operating systems) to determine whether paging is occurring on your system.

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.

NUMDB

The behavior of DB2 Connect was unaffected by the **NUMDB** configuration parameter in previous versions, however, this changed starting with Version 8. This parameter indicates the maximum number of databases the clients can connect to through the DB2 Connect server. More specifically, the maximum number of different database aliases that can be catalogued on DB2 Connect server.

Other DB2 Connect parameters

The **AGENTPRI** and **MAXAGENTS** are deprecated in Version 9.5

Commands to update the value for **MAXAGENTS** will continue to work so that existing applications are not broken, but the values will be ignored. The parameter name will not appear in any configuration lists. In the past, the total number of agents allowed to be created on a given DB2 partition was controlled through the **MAXAGENTS** configuration parameter. You now have the ability to automate the configuration of agents.

By default, **NUM_POOLAGENTS** will be set to AUTOMATIC with a value of 100 as the default. Also by default, **MAX_COORDAGENTS** will be set to AUTOMATIC with a value of 200 as the default.

To send accounting strings from your client applications to the DB2 Connect server, use the API-specific means for setting accounting information. The API-specific means perform faster than setting the DB2ACCOUNT environment variable.

IBM Data Server Driver for JDBC and SQLJ
com.ibm.db2.jcc.DB2BaseDataSource.clientAccountingInformation property

DB2 .NET Data Provider
DB2Connection.ClientAccountingInformation property

CLI/ODBC
ClientAcctStr CLI/ODBC configuration keyword

Embedded SQL (C, C++, and COBOL)
sqlesact function

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.

Host database tuning

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

You might be able to improve performance by using the uncommitted read (UR) or no commit (NC) bind options, where available, to avoid journaling.

Note: When using UR, unjournaled 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 might 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.

Network tuning considerations

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.

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.

A summary of network performance tuning methods is provided in Table 17 on page 116.

Table 17. Network performance tuning methods

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.
Buffers	RUSIZE	Optimum size is 4096.	Setting RUSIZE and RQRIOBLK to same size might give the best performance.
Buffers	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.
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.
TCP Settings	Send/Receive Space Sizes	Should be 64K for both.	Default is only 8192 for Windows. Can be set in the Windows registry.

System resources contention

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?

DB2 Connect performance troubleshooting

If DB2 Connect users are experiencing long response times during large queries from host or System i 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 System i 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. Ensure the maximum RU size defined in the IBMRDB mode definition is set to a suitable value. It is recommended that the size is 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 might be a limiting factor.

Tuning DB2 for OS/390 and z/OS

You can optimize inactive thread processing in OS/390 and z/OS. In V5, you are allowed up to 25,000 concurrently connected clients. 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.

Increasing DB2 Connect data transfer rates

In addition to blocking of rows for a query result set, DB2 for OS/390 and z/OS can also return multiple such query blocks in response to an OPEN or FETCH request to a remote client, such as DB2 Connect. Instead of the client repeatedly sending requests to the DB2 for OS/390 and z/OS server requesting one block of row data at a time, the client can now optionally request that the server send back some number of query blocks in addition to the one that it will always send back. Such additional query blocks are called extra query blocks.

As such, this new feature allows the client to minimize the number of network line turnarounds, which constitute a major cost to network performance. The decrease in the number of requests sent by the client to the server for query blocks translates into a significant performance boost. This performance boost is due to the fact that switching between a send and receive is an expensive operation performance-wise. DB2 Connect can now exploit this performance enhancement by requesting extra query blocks from a DB2 for OS/390 and z/OS server by default.

To fully take advantage of the return of extra query blocks (each of which can be up to 32K bytes long) for the preferred network protocol of TCP/IP, window scaling extensions have been enabled as architected under RFC-1323 in DB2 Connect. This feature that allows TCP/IP to dynamically adjust the send and receive window sizes to accommodate the potentially large amounts of data returned by way of the extra query blocks efficiently.

Extra query block

Extra query block support on servers with DB2 Universal Database (UDB) for OS/390 and z/OS Version 7 or later is configured via the EXTRA BLOCKS SRV parameter on the DB2 DDF installation panel. This support is configured by way of controlling the maximum number of extra query blocks that DB2 can send back to a client for a request. You can set this parameter to a value between 0 and 100. Setting the parameter value to 0 disables the return of extra query blocks. The default value of 100 should always be used to get the most benefit out of this feature, barring any idiosyncrasies in the network that would render this setting less than ideal.

On the client side, where the application accesses DB2 for z/OS either directly through a co-located DB2 Connect installation, or through a separate DB2 Connect server installation, there are various means for activating the corresponding DB2 Connect support on a per cursor or statement basis:

- The use of a query rowset size for a cursor
- The use of the 'OPTIMIZE for N ROWS' clause on the select statement associated with a cursor
- The use of the 'FETCH FIRST N ROWS ONLY' clause on the select statement associated with a cursor

DB2 Connect can enable extra query block support using different SQL APIs:

Embedded SQL

- The user can invoke extra query block support for a query by specifying either the 'OPTIMIZE for N ROWS' clause, or the 'FETCH FIRST N ROWS ONLY' clause, or both on the select statement itself.
- With the 'OPTIMIZE for N ROWS' clause, DB2 for OS/390 and z/OS will attempt to block the desired number of rows to return to DB2 Connect, subject to the EXTRA BLOCKS SRV DDF installation parameter setting. The application can choose to fetch beyond N rows as DB2 for z/OS does not limit the total number of rows that could ultimately be returned for the query result set to N.
- The 'FETCH FIRST N ROWS ONLY' clause works similarly, except that the query result set is limited to N rows by DB2 for OS/390 and z/OS. Fetching beyond N rows would result in SQL code +100 (end of data).

CLI/ODBC

- The user can invoke extra query block support for a query through its SQL_MAX_ROWS statement attribute.
- The 'FETCH FIRST N ROWS ONLY' clause is used instead for a DB2 UDB for OS/390 and z/OS 7.1 or later server.
  - For Version 7, the query result set is limited to N rows by DB2 for OS/390 and z/OS. Fetching beyond N rows would result in SQL_NO_DATA_FOUND.
  - For Version 8 or later, the CLI ensures that only the first N rows are returned to the application via the client Cursor Manager.

JDBC The user can invoke extra query block support for a query through the setMaxRows method. Similar to the CLI/ODBC enablement, DB2 Connect will tag on the 'OPTIMIZE for N ROWS' clause for a DB2 for OS/390 and z/OS 6.x server. DB2 Connect will also tag the 'FETCH FIRST N ROWS ONLY' clause for a DB2 for z/OS 7.1 or above server.

RFC-1323 Window scaling

Window scaling is supported on all Windows, Linux, and UNIX platforms that support the RFC-1323 extensions for TCP/IP. You can enable this feature on DB2 for Windows, Linux, or UNIX using the DB2 registry variable DB2SORCVBUF. To turn window scaling on, this registry variable should be set to any value above 64K. For example, on DB2 for Windows, Linux, or UNIX, you can issue db2set DB2SORCVBUF =65537.

The maximum send and receive buffer sizes are dependent on the specific operating system. To ensure that buffer sizes configured have been accepted, the

user can set the database manager configuration parameter DIAGLEVEL to 4 (informational) and check the administration notification log file for messages.

For window scaling to take effect it must be enabled on both ends of a connection; on both the workstation and the host, either directly through the operating system TCP/IP stack, or indirectly through the DB2 product. For instance, for DB2 for z/OS, window scaling can currently only be activated through the operating system by setting TCPRCVBUFRSIZE to any value above 64K. If you are using a remote IBM data server client to access a host or System i DB2 database through a DB2 Connect server workstation, you can enable window scaling on the client as well. By the same token, you can also enable window scaling between a remote IBM data server client and a workstation DB2 server when no host or System i DB2 database is involved.

While window scaling is designed to enhance network performance, it is important to note that the expected network performance improvement does not always materialize. Interaction among factors such as the frame size used for the ethernet or token ring LAN adapter, the IP MTU size, and other settings at routers throughout the communication link could even result in performance degradation once window scaling has been enabled. Therefore, by default, window scaling is disabled with both the send and receive buffers set to 64K.

You should be prepared to assess the impact of turning on window scaling and perform any necessary adjustments to the network. For an introduction to tuning the network for improved network performance, refer to http://www.networking.ibm.com/.

Host data conversion

When information is transferred between different environments (such as Intel [Windows], IEEE [Linux and UNIX operating systems], zSeries [VM, VSE, z/OS], System i [OS/400]), numeric data types (such as decimal, integer, floating point) might need to be converted. This conversion can affect performance.

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 might 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 System i 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 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 System i 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 network to another 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.

Chapter 20. CLI/ODBC application performance tuning

CLI/ODBC is an SQL application programming interface that can be called by your database applications. CLI functions invoke DB2 stored procedures which, in turn, access the system catalog tables.

Some applications use ODBC APIs to gather metadata information that is used in further processing. The ten metadata API calls that can be made are:

- `SQLTables`
- `SQLColumns`
- `SQLSpecialcolumns`
- `SQLStatistics`
- `SQLPrimarykeys`
- `SQLForeignkeys`
- `SQLTablePrivileges`
- `SQLColumnPrivileges`
- `SQLProcedures`
- `SQLProcedureColumns`

Certain CLI/ODBC applications that use the metadata APIs listed above might query all of the objects within the database. For example, an `SQLTables` call requests metadata for all the tables in the database. On a large system, such requests can result in a lot of network traffic, take a considerable amount of time and consume a considerable amount of server resources.

Several CLI/ODBC initialization keywords 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`
- `GrantorList`
- `GranteeList`

Part 5. Troubleshooting

Chapter 21. Troubleshooting

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

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

Gathering relevant information

Troubleshooting 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 one or more clients and the DB2 Connect Server (gateway); between the DB2 Connect gateway and the host or System i database server; or between theDB2 Connect Personal Edition and the host or System i database server?
- What can you determine by the content of the message and the tokens returned in the message?
- Will using diagnostic tools such as db2trc, db2pd, or db2support 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 operating 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 operating systems, did you edit the nodelock file and enter the password that IBM supplied?

4. *Were the host or System i database server and workstation communications configured properly?*
   - There are three configurations that must be considered:
     a. The host or System i database server configuration identifies the application requester to the server. The host or System i 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 System i 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 that 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 System i 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 System i server database management system to use the host or System i 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 (CLP) to issue SQL statements against a host or System i database server, are you unsuccessful?*
   - Did you follow the procedure to bind the CLP to the host or System i 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 fix pack 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 fix pack and load it *after* installing the feature.
2. *Has this error occurred before?*
   - Are there any documented resolutions to previous error conditions?
   - Who were the participants and can they provide insight into a possible course of action?

3. *Have you explored using communications software commands that return information about the network?*
   - TCP/IP might have 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 might provide a common base.
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 might have been reached. If another client disconnects from the server, is the client who was previously unable 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 might be successful, but the authorization might not have been 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 might block communication between the client and the server. For example, when using TCP/IP, ensure that you can PING the remote host.

Diagnostic tools

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

- All diagnostic data including dump files, trap files, error logs, notification files, and alert logs are found in the path specified by the diagnostic data directory path (**diagpath**) database manager configuration parameter:

  If the value for this configuration parameter is null, the diagnostic data is written to one of the following directories or folders:

  – For Linux and UNIX environments: INSTHOME/sqllib/db2dump, where *INSTHOME* is the home directory of the instance.

  – For supported Windows environments:

    - If the **DB2INSTPROF** environment variable is not set then x:\SQLLIB\DB2INSTANCE is used where x:\SQLLIB is the drive reference and the directory specified in the **DB2PATH** registry variable, and the value of **DB2INSTANCE** has the name of the instance.

      Note: The directory does not have to be named SQLLIB.

    - If the **DB2INSTPROF** environment variable is set then x:\DB2INSTPROF\DB2INSTANCE is used where **DB2INSTPROF** is the name of the instance profile directory and **DB2INSTANCE** is the name of the instance (by default, the value of **DB2INSTDEF** on Windows 32-bit operating systems).

- For Windows operating systems, you can use the Event Viewer to view the administration notification log.

- The available diagnostic tools that can be used include **db2trc**, **db2pd** and **db2support**.
- For Linux and UNIX operating systems, the **ps** command, which returns process status information about active processes to standard output.
- For UNIX operating 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.

Chapter 22. DB2 traces within DB2 Connect

Tracing actions and operations as they are happening within your environment can provide useful information when troubleshooting a problem. You can obtain, dump, and format a trace taken within the DB2 database server product. The trace facility is provided as part of the DB2 database server product.

Obtaining a DB2 trace using db2trc

The **db2trc** command controls the trace facility provided with DB2. The trace facility records information about operations and formats this information into a readable form.

Keep in mind that there is added overhead when a trace is running so enabling the trace facility might impact your system's performance.

In general, DB2 Support and development teams use DB2 traces for troubleshooting. You might run a trace to gain information about a problem that you are investigating, but its use is rather limited without knowledge of the DB2 source code.

Nonetheless, it is important to know how to correctly turn on tracing and how to dump trace files, just in case you are asked to obtain them.

Note: You will need one of SYSADM, SYSCTRL or SYSMAINT authority to use db2trc

To get a general idea of the options available, execute the db2trc command without any parameters:

```
C:\>db2trc
Usage: db2trc (chg|clr|dmp|flw|fmt|inf|off|on) options
```

For more information about a specific db2trc command parameter, use the `-u` option. For example, to see more information about turning the trace on, execute the following command:

```
db2trc on -u
```

This will provide information about all of the additional options (labeled as "facilities") that can be specified when turning on a DB2 trace.

When turning trace on, the most important option is `-L`. This specifies the size of the memory buffer that will be used to store the information being traced. The buffer size can be specified in either bytes or megabytes. (To specify megabytes append either "M" or "m" after the value). The trace buffer size must be a power of two megabytes. If you specify a size that does not meet this requirement, the buffer size will automatically be rounded down to the nearest power of two.

If the buffer is too small, information might be lost. By default only the most recent trace information is kept if the buffer becomes full. If the buffer is too large, it might be difficult to send the file to the DB2 support team.

If tracing an operation that is relatively short (such as a database connection), a size of approximately 8MB is usually sufficient:

```
C:\> db2trc on -l 8M
Trace is turned on
```

However, if you are tracing a larger operation or if a lot of work is going on at the same time, a larger trace buffer might be required.

On most platforms, tracing can be turned on at any time and works as described above. However, there are certain situations to be aware of:

1. On multiple database partition systems, you must run a trace for each physical (as opposed to logical) database partition.
2. On HP-UX, Linux and Solaris platforms, if the trace is turned off after the instance has been started, a very small buffer will be used the next time the trace is started regardless of the size specified. For example, yesterday you turned trace on by using db2trc on -l 8m, then collected a trace, and then turned the trace off (db2trc off). Today you wish to run a trace with the memory buffer set for 32 megabytes (db2trc on -l 32m) without bringing the instance down and restarting. You will find that in this case trace will only get a small buffer. To effectively run a trace on these platforms, turn the trace on before starting the instance with the size buffer you need and "clear" the buffer as necessary afterwards.

Dumping a DB2 trace file

Once the trace facility has been enabled using the on option, all subsequent work done by the instance will be traced.

While the trace is running, you can use the clr option to clear out the trace buffer. All existing information in the trace buffer will be removed.

```
C:\>db2trc clr
Trace has been cleared
```

Once the operation being traced has finished, use the dmp option followed by a trace file name to dump the memory buffer to disk. For example:

```
C:\>db2trc dmp trace.dmp
Trace has been dumped to file
```

The trace facility will continue to run after dumping the trace buffer to disk. To turn tracing off, use the off option:

```
C:\>db2trc off
Trace is turned off
```

Formatting a DB2 trace file

The dump file created by the command db2trc dmp is in binary format and is not readable.

To verify that a trace file can be read, format the binary trace file to show the flow control and send the formatted output to a null device. The following example shows the command to perform this task:

```
db2trc flw example.trc nul
```

where example.trc is a binary file that was produced using the dmp option.

The output for this command will explicitly tell you if there is a problem reading the file, and whether or not the trace was wrapped.

At this point, the dump file could be sent to DB2 Support. They would then format it based on your DB2 service level. However, you might sometimes be asked to format the dump file into ASCII format before sending it. This is accomplished via the flw and fmt options. You must provide the name of the binary dump file along with the name of the ASCII file that you want to create:

```
C:\>db2trc flw trace.dmp trace.flw
C:\Temp>db2trc flw trace.dmp trace.flw
Total number of trace records     : 18854
Trace truncated                   : NO
Trace wrapped                     : NO
Number of trace records formatted : 1513 (pid: 2196 tid 2148 node: -1)
Number of trace records formatted : 100 (pid: 1568 tid 1304 node: 0)
...

C:\>db2trc fmt trace.dmp trace.fmt
C:\Temp>db2trc fmt trace.dmp trace.fmt
Trace truncated                   : NO
Trace wrapped                     : NO
Total number of trace records     : 18854
Number of trace records formatted : 18854
```

If this output indicates ″Trace wrapped″ is ″YES″, then this means that the trace buffer was not large enough to contain all of the information collected during the trace period. A wrapped trace might be okay depending on the situation. If you are interested in the most recent information (this is the default information that is maintained, unless the -i option is specified), then what is in the trace file might be sufficient. However, if you are interested in what happened at the beginning of the trace period or if you are interested in everything that occurred, you might want to redo the operation with a larger trace buffer.

There are options available when formatting a binary file into a readable text file. For example, you can use `db2trc fmt -xml trace.dmp trace.fmt` to convert the binary data and output the result into an xml parsable format. Additional options are shown in the detailed description of the trace command (db2trc).

Another thing to be aware of is that on Linux and UNIX operating systems, DB2 will automatically dump the trace buffer to disk when it shuts the instance down due to a severe error. Thus if tracing is enabled when an instance ends abnormally, a file will be created in the diagnostic directory and its name will be db2trdmp.###, where ### is the database partition number. This does not occur on Windows platforms. You have to dump the trace manually in those situations.

To summarize, the following is an example of the common sequence of db2trc commands:

```
db2trc on -l 8M
db2trc clr
<Execute problem recreation commands>
db2trc dump db2trc.dmp
db2trc off
db2trc flw db2trc.dmp <filename>.flw
db2trc fmt db2trc.dmp <filename>.fmt
db2trc fmt -c db2trc.dmp <filename>.fmtc
```

Chapter 23. DRDA trace files

Before analyzing DRDA traces, you need to understand that DRDA is an open standard for the definition of data and communication structures. For example, DRDA comprises a set of rules about how data should be organized for transmission and how communication of that information should occur. These rules are defined in the following reference manuals:

- DRDA V3 Vol. 1: Distributed Relational Database Architecture
- DRDA V3 Vol. 2: Formatted Data Object Content Architecture
- DRDA V3 Vol. 3: Distributed Data Management Architecture

PDF versions of these manuals are available on www.opengroup.org.

The **db2drdat** utility records the data interchanged between a DRDA Application Requestor (AR) and a DB2 DRDA Application Server (AS) (for example between DB2 Connect and a host or Series i database server).

Trace utility

The db2drdat utility records the data interchanged between the DB2 Connect server (on behalf of the IBM data server client) and the host or System i database server.

As a database administrator (or application developer), you might find it useful to understand how this flow of data works, because this knowledge can help you determine the origin of a particular problem. Suppose you found yourself in the following situation: you issue a `CONNECT TO` database statement for a host or System i 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 System i database server management system, you might 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 `db2drdat` lists the data streams exchanged between the DB2 Connect workstation and the host or System i database server management system. Data sent to the host or System i database server is labeled SEND BUFFER and data received from the host or System i 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 System i database server. 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.

Trace output

The db2drdat utility writes the following information to *tracefile*:

- -r
  - Type of DRDA reply/object
  - Receive buffer
- -s
  - Type of DRDA request
  - Send buffer
- -c
  - SQLCA
- TCP/IP error information
  - Receive function return code
  - Severity
  - Protocol used
  - API used
  - Function
  - Error number.

Note:

1. A value of zero for the exit code indicates that the command completed successfully, and a non-zero value indicates that it did not.
2. The fields returned vary based on the API used.
3. The fields returned vary based on the platform on which DB2 Connect is running, even for the same API.
4. If the db2drdat command 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.

Trace output file analysis

The following information is captured in a db2drdat trace :

- 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 System i database server CCSID(s)
- The host or System i 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 System i database server management system. It sends these commands as a result of a `CONNECT TO` database command. The next buffer contains the reply that DB2 Connect received from the host or

System i database server management system. It contains an Exchange Server Attributes Reply Data (EXCSATRD) and an Access RDB Reply Message (ACCRDBRM).

EXCSAT

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'9481A292' (coded in CCSID 500) are translated to *mask* 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 System i 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` padded with blanks followed by `000C50CC`. On a Linux or UNIX IBM data server client, this value can be correlated with the ps command, which returns process status information about active processes to standard output.

ACCRDB

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'E2E3D3C5C3F1' are translated to STLEC1 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'.

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'0333', which is 819.

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 also present in the ACCRDB command. In this example, the CCSIDDBC is X'04B0', which is 1200, and the CCSIDMBC is X'0333', which is 819, respectively.

EXCSATRD and ACCRDBRM

CCSID values are also returned from the host or System i 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 two CCSID values for the host or System i database server system. The values are 1208 (for both single-byte and mixed byte characters) and 1200 (for double-byte characters).

If DB2 Connect does not recognize the code page coming back from the host or System i database server, SQLCODE -332 will be returned to the user with the source and target code pages. If the host or System i 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 -332 for the user.

The ACCRDBRM also contains the parameter PRDID (Product-specific Identifier, with code point X'112E'). The value is X'C4E2D5F0F8F0F1F5' which is DSN08015 in EBCDIC. According to standards, DSN is DB2

Universal Database for z/OS and OS/390. The version number is also indicated. ARI is DB2 Server for VSE & VM, SQL is DB2 database or DB2 Connect, and QSQ is DB2 for i5/OS.

Trace output file samples

The following figures show sample output illustrating some DRDA data streams exchanged between DB2 Connect workstations and a host or System i database server. From the user's viewpoint, a `CONNECT TO` database command has been issued using the command line processor (CLP).

Figure 13 on page 139 uses DB2 Connect Enterprise Edition Version 9.1 and DB2 Universal Database (UDB) for z/OS Version 8 over a TCP/IP connection.

```
1 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 0 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 233

2 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 19532 probe 1177
 bytes 250

 SEND BUFFER(AR):

           EXCSAT RQSDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  00C3D041000100BD 1041007F115E8482   ...A.....A...^..  .C}........".;db
  0010  F282974040404040 4040404040404040   ...@@@@@@@@@@@@@  2bp
  0020  4040F0F0F0C3F5F0 C3C3F0F0F0000000   @@.............    000C50CC000...
  0030  0000000000000000 0000000000000000   ................  ................
  0040  0000000000000000 000000000060F0F0   ............`..  ............-00
  0050  F0F1A2A495404040 4040404040404040   .....@@@@@@@@@@@  01sun
  0060  4040404040404040 4040404040404040   @@@@@@@@@@@@@@@@
  0070  C4C5C3E5F8404040 F0A2A49540404040   .....@@@....@@@@  DECV8   0sun
  0080  4040404040404040 4000181404140300   @@@@@@@@@.......           .......
  0090  0724070008147400 05240F0008144000   .$....t..$....@.  ..............
  00A0  08000E1147D8C4C2 F261C1C9E7F6F400   ....G....a......  .....QDB2/AIX64.
  00B0  08116D9481A29200 0C115AE2D8D3F0F9   ..m.......Z.....  .._mask...]SQL09
  00C0  F0F0F0                              ...               000

           ACCSEC RQSDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0026D00100020020 106D000611A20003   .&..... .m......  ..}....._...s..
  0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
  0020  404040404040                        @@@@@@

3 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110546200 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 105

4 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110549755 probe 1178
 bytes 122

 RECEIVE BUFFER(AR):

           EXCSATRD OBJDSS                  (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0059D04300010053 1443000F115EE5F8   .Y.C...S.C...^..  ..}.........;V8
  0010  F1C14BE2E3D3C5C3 F100181404140300   ..K.............  1A.STLEC1.......
  0020  0724070007147400 05240F0007144000   .$....t..$....@.  ..............
  0030  0700081147D8C4C2 F20014116DE2E3D3   ....G.......m...  .....QDB2..._STL
  0040  C5C3F14040404040 4040404040000C11   ...@@@@@@@@@@...  EC1          ...
  0050  5AC4E2D5F0F8F0F1 F5                Z........         ]DSN08015

           ACCSECRD OBJDSS                  (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0010D0030002000A 14AC000611A20003   ..............  ..}.........s..

5 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110656806 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 233
```

Figure 13. Example of Trace Output (TCP/IP connection)

```
6 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110659711 probe 1177
 bytes 250

 SEND BUFFER(AR):

           SECCHK RQSDSS                    (ASCII)           (EBCDIC)
          0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  003CD04100010036 106E000611A20003   .<.A...6.n......  ..}......>...s..
  0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
  0020  4040404040040000C 11A1D9858799F485   @@@@@@..........        ....Regr4e
  0030  A599000A11A09585 A6A39695           ............      vr....newton

           ACCRDB RQSDSS                    (ASCII)           (EBCDIC)
          0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  00ADD001000200A7 20010006210F2407   ........ ...!.$.  ..}....x........
  0010  00172135C7F9F1C1 F0C4F3C14BD7C1F8   ..!5........K...  ....G91A0D3A.PA8
  0020  F806030221064600 162110E2E3D3C5C3   ....!.F..!......  8..........STLEC
  0030  F140404040404040 4040404040000C11   .@@@@@@@@@@@@...  1            ...
  0040  2EE2D8D3F0F9F0F0 F0000D002FD8E3C4   ............/...  .SQL09000....QTD
  0050  E2D8D3C1E2C30016 00350006119C0333   .........5.....3  SQLASC..........
  0060  0006119D04B00006 119E0333003C2104   ...........3.

7 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 259908001 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 176

8 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 259911584 probe 1178
 bytes 193

 RECEIVE BUFFER(AR):

           SECCHKRM RPYDSS                  (ASCII)           (EBCDIC)
          0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0015D0420001000F 1219000611490000   ...B.........I..  ..}.............
  0010  000511A400                          .....             ...u.

           ACCRDBRM RPYDSS                  (ASCII)           (EBCDIC)
          0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  009BD00200020095 2201000611490000   ........"....I..  ..}....n........
  0010  000D002FD8E3C4E2 D8D3F3F7F0000C11   .../............  ....QTDSQL370...
  0020  2EC4E2D5F0F8F0F1 F500160035000611   ............5...  .DSN08015.......
  0030  9C04B80006119E04 B80006119D04B000   ................  ................
  0040  0C11A0D5C5E6E3D6 D540400006212524   .........@@..!%$  ...NEWTON  .....
  0050  34001E244E000624 4C00010014244D00   4..$N..$L....$M.  ....+...<.....(.
  0060  06244FFFFF000A11 E8091E768301BE00   .$O........v....  ..!.....Y...c...
  0070  2221030000000005 68B3B8C7F9F1C1F0   "!......h.......  ...........G91A0
  0080  C4F3C1D7C1F8F840 4040400603022106   .......@@@...!.  D3APA88    .....
  0090  46000A11E8091E76 831389            F......v...       ....Y...c.i

9 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 364420503 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 10
```

Figure 14. Example of Trace Output (TCP/IP connection) continued

```
10 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 364440751 probe 1177
 bytes 27

 SEND BUFFER(AR):

           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               ........ .        ..}.......
11 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 475009631 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 54
12 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 475014579 probe 1178
 bytes 71

 RECEIVE BUFFER(AR):

           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  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
  0020  4040404040400005 211501             @@@@@@..!..             .....

           SQLCARD OBJDSS                   (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000BD00300010005 2408FF             ........$..       ..}........
13 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 721710319 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 126
14 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 721727276 probe 1177
 bytes 143

 SEND BUFFER(AR):

           EXCSQLIMM RQSDSS                 (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0053D0510001004D 200A00442113E2E3   .S.Q...M ..D!...  ..}....(......ST
  0010  D3C5C3F140404040 4040404040404040   ....@@@@@@@@@@@@  LEC1
  0020  D5E4D3D3C9C44040 4040404040404040   ......@@@@@@@@@@  NULLID
  0030  4040E2D8D3C3F2C6 F0C1404040404040   @@........@@@@@@    SQLC2F0A
  0040  4040404041414141 41484C5600CB0005   @@@@AAAAAAHLV....      ......<.....
  0050  2105F1                              !..               ..1

           SQLSTT OBJDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  002BD00300010025 2414000000001B64   .+.....%$......d  ..}.............
  0010  656C657465206672 6F6D206464637375   elete from ddcsu  .%......?_......
  0020  73312E6D79746162 6C65FF             s1.mytable.       ..._`./.%..
15 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 832901261 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 102
```

Figure 15. Example of Trace Output (TCP/IP connection) continued

```
16 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 832906528 probe 1178
 bytes 119

 RECEIVE BUFFER(AR):

           SQLCARD OBJDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0066D00300010060 240800FFFFFF3434   .f.....`$.....44  ..}....-........
  0010  3237303444534E58 4F544C2000FFFFFE   2704DSNXOTL ....  ......+.!.<.....
  0020  0C00000000000000 00FFFFFFFF000000   ................  ................
  0030  0000000000572020 2057202020202020   .....W   W       ................
  0040  001053544C454331 2020202020202020   ..STLEC1          ....<...........
  0050  2020000F44444353 5553312E4D595441     ..DDCSUS1.MYTA  ............(...
  0060  424C450000FF                        BLE...            .<....
17 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 833156953 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 10
18 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 833159843 probe 1177
 bytes 27

 SEND BUFFER(AR):

           RDBRLLBCK RQSDSS                  (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000AD00100010004 200F               ........ .        ..}.......
19 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 943302832 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 54
20 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 943306288 probe 1178
 bytes 71

 RECEIVE BUFFER(AR):

           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  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
  0020  4040404040400005 211502             @@@@@@..!..             .....

           SQLCARD OBJDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000BD00300010005 2408FF             ........$..       ..}........
```

Figure 16. Example of Trace Output (TCP/IP connection) continued

Subsequent buffer information for DRDA traces

You can analyze subsequent send and receive buffers for additional information. The next request contains a commit. The commit command instructs the host or System i database server management system to commit the current unit of work. The fourth buffer is received from the host or System i 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, trace entry 12 contains a null SQLCA, indicated by DDM code point X'2408' followed by X'FF'. A null SQLCA (X'2408FF') indicates success (SQLCODE 0).

Figure 13 on page 139 shows an example of a receive buffer containing an error SQLCA at trace entry 16.

Part 6. Messages

Chapter 24. Common DB2 Connect problems

This topic 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, usually in the form of a sample message listing.
- A suggested solution, indicating the probable cause of the error. In some cases, more than one suggested solution might be provided.

SQL0965 or SQL0969

Symptom

Messages SQL0965 and SQL0969 can be issued with a number of different return codes from DB2 for i5/OS, DB2 for z/OS, 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 or i5/OS database cannot be translated. Correct the problem, based on the error code, then resubmit the failing command.

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 server. There might have been a successful client connection previously.

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

```
2001-05-30-14.09.55.321092   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 server for remote clients, is having trouble handling one or more client communication protocols. These protocols can be TCP/IP and others, and usually the message indicates that one of the communications protocols defined to DB2 Connect is not configured properly.

Often the cause might 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 or nname).

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 might still appear to exist on the client, and it might become possible to restore the connection without further intervention by issuing the commands shown below.

Most likely, one of the clients connecting to the DB2 Connect server still has a handle on the TCP/IP port. On each client machine that is connected to the DB2 Connect server, enter the following commands:

```
db2 terminate
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. Run the db2support command before contacting service.

SQL30060

Symptom

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

Solution

When connecting to DB2 for OS/390 and z/OS, the Communications Database (CDB) tables have not been updated properly.

SQL30061

Symptom

Connecting to the wrong host or System i database server location - no target database can be found.

Solution

The wrong server database name might 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 z/OS and 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.

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 server
```

To connect to the database you then issue:

```
db2 connect to <alias> user <user_name> using <password>
```

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 server. It can also occur when connecting from the DB2 Connect server to a host or System i database server.

1. The DB2COMM profile variable might be set incorrectly on the DB2 Connect server. Check this. For example, the command db2set db2comm=tcpip should appear in sqllib/db2profile when running DB2 Enterprise Server Edition on AIX.
2. There might be a mismatch between the TCP/IP service name and port number specifications at the IBM data server client and the DB2 Connect server. Verify the entries in the TCP/IP services files on both machines.
3. Check that DB2 is started on the DB2 Connect server. 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:

   ```
   2001-02-03-12.41.04.861119   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 might 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 might be required.

SQL30082 RC=24 During CONNECT

Symptom

SQLCODE -30082 The username or the password supplied is incorrect.

Solution

Ensure that the correct password is provided on the CONNECT statement if necessary. Password not available to send to the target server database. A password has to be sent from the IBM data server 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.

Part 7. Appendixes

Appendix A. Overview of the DB2 technical information

DB2 technical information is available through the following tools and methods:

- DB2 Information Center
  - Topics (Task, concept and reference topics)
  - Help for DB2 tools
  - Sample programs
  - Tutorials
- DB2 books
  - PDF files (downloadable)
  - PDF files (from the DB2 PDF DVD)
  - printed books
- Command line help
  - Command help
  - Message help

Note: The DB2 Information Center topics are updated more frequently than either the PDF or the hard-copy books. To get the most current information, install the documentation updates as they become available, or refer to the DB2 Information Center at ibm.com®.

You can access additional DB2 technical information such as technotes, white papers, and IBM Redbooks publications online at ibm.com. Access the DB2 Information Management software library site at http://www.ibm.com/software/data/sw-library/.

Documentation feedback

We value your feedback on the DB2 documentation. If you have suggestions for how to improve the DB2 documentation, send an email to db2docs@ca.ibm.com. The DB2 documentation team reads all of your feedback, but cannot respond to you directly. Provide specific examples wherever possible so that we can better understand your concerns. If you are providing feedback on a specific topic or help file, include the topic title and URL.

Do not use this email address to contact DB2 Customer Support. If you have a DB2 technical issue that the documentation does not resolve, contact your local IBM service center for assistance.

DB2 technical library in hardcopy or PDF format

The following tables describe the DB2 library available from the IBM Publications Center at www.ibm.com/shop/publications/order. English DB2 Version 9.5 manuals in PDF format and translated versions can be downloaded from www.ibm.com/support/docview.wss?rs=71&uid=swg2700947.

Although the tables identify books available in print, the books might not be available in your country or region.

The form number increases each time a manual is updated. Ensure that you are reading the most recent version of the manuals, as listed below.

Note: The DB2 Information Center is updated more frequently than either the PDF or the hard-copy books.

Table 18. DB2 technical information

Name	Form Number	Available in print
Administrative API Reference	SC23-5842-01	Yes
Administrative Routines and Views	SC23-5843-01	No
Call Level Interface Guide and Reference, Volume 1	SC23-5844-01	Yes
Call Level Interface Guide and Reference, Volume 2	SC23-5845-01	Yes
Command Reference	SC23-5846-01	Yes
Data Movement Utilities Guide and Reference	SC23-5847-01	Yes
Data Recovery and High Availability Guide and Reference	SC23-5848-01	Yes
Data Servers, Databases, and Database Objects Guide	SC23-5849-01	Yes
Database Security Guide	SC23-5850-01	Yes
Developing ADO.NET and OLE DB Applications	SC23-5851-01	Yes
Developing Embedded SQL Applications	SC23-5852-01	Yes
Developing Java Applications	SC23-5853-01	Yes
Developing Perl and PHP Applications	SC23-5854-01	No
Developing User-defined Routines (SQL and External)	SC23-5855-01	Yes
Getting Started with Database Application Development	GC23-5856-01	Yes
Getting Started with DB2 installation and administration on Linux and Windows	GC23-5857-01	Yes
Internationalization Guide	SC23-5858-01	Yes
Message Reference, Volume 1	GI11-7855-00	No
Message Reference, Volume 2	GI11-7856-00	No
Migration Guide	GC23-5859-01	Yes
Net Search Extender Administration and User's Guide	SC23-8509-01	Yes
Partitioning and Clustering Guide	SC23-5860-01	Yes
Query Patroller Administration and User's Guide	SC23-8507-00	Yes
Quick Beginnings for IBM Data Server Clients	GC23-5863-01	No

Table 18. DB2 technical information (continued)

Name	Form Number	Available in print
Quick Beginnings for DB2 Servers	GC23-5864-01	Yes
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC23-8508-01	Yes
SQL Reference, Volume 1	SC23-5861-01	Yes
SQL Reference, Volume 2	SC23-5862-01	Yes
System Monitor Guide and Reference	SC23-5865-01	Yes
Troubleshooting Guide	GI11-7857-01	No
Tuning Database Performance	SC23-5867-01	Yes
Visual Explain Tutorial	SC23-5868-00	No
What's New	SC23-5869-01	Yes
Workload Manager Guide and Reference	SC23-5870-01	Yes
pureXML Guide	SC23-5871-01	Yes
XQuery Reference	SC23-5872-01	No

Table 19. DB2 Connect-specific technical information

Name	Form Number	Available in print
Quick Beginnings for DB2 Connect Personal Edition	GC23-5839-01	Yes
Quick Beginnings for DB2 Connect Servers	GC23-5840-01	Yes
DB2 Connect User's Guide	SC23-5841-01	Yes

Table 20. Information Integration technical information

Name	Form Number	Available in print
Information Integration: Administration Guide for Federated Systems	SC19-1020-01	Yes
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-02	Yes
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-01	No
Information Integration: SQL Replication Guide and Reference	SC19-1030-01	Yes
Information Integration: Introduction to Replication and Event Publishing	SC19-1028-01	Yes

Ordering printed DB2 books

If you require printed DB2 books, you can buy them online in many but not all countries or regions. You can always order printed DB2 books from your local IBM representative. Keep in mind that some softcopy books on the *DB2 PDF Documentation* DVD are unavailable in print. For example, neither volume of the *DB2 Message Reference* is available as a printed book.

Printed versions of many of the DB2 books available on the DB2 PDF Documentation DVD can be ordered for a fee from IBM. Depending on where you are placing your order from, you may be able to order books online, from the IBM Publications Center. If online ordering is not available in your country or region, you can always order printed DB2 books from your local IBM representative. Note that not all books on the DB2 PDF Documentation DVD are available in print.

Note: The most up-to-date and complete DB2 documentation is maintained in the DB2 Information Center at http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

To order printed DB2 books:

- To find out whether you can order printed DB2 books online in your country or region, check the IBM Publications Center at http://www.ibm.com/shop/publications/order. You must select a country, region, or language to access publication ordering information and then follow the ordering instructions for your location.
- To order printed DB2 books from your local IBM representative:
  1. Locate the contact information for your local representative from one of the following Web sites:
     - The IBM directory of world wide contacts at www.ibm.com/planetwide
     - The IBM Publications Web site at http://www.ibm.com/shop/publications/order. You will need to select your country, region, or language to the access appropriate publications home page for your location. From this page, follow the "About this site" link.
  2. When you call, specify that you want to order a DB2 publication.
  3. Provide your representative with the titles and form numbers of the books that you want to order. For titles and form numbers, see "DB2 technical library in hardcopy or PDF format" on page 153.

Displaying SQL state help from the command line processor

DB2 returns an SQLSTATE value for conditions that could be the result of an SQL statement. SQLSTATE help explains the meanings of SQL states and SQL state class codes.

To invoke SQL state help, open the command line processor and enter:

```
? sqlstate or ? class code
```

where *sqlstate* represents a valid five-digit SQL state and *class code* represents the first two digits of the SQL state.

For example, `? 08003` displays help for the 08003 SQL state, and `? 08` displays help for the 08 class code.

Accessing different versions of the DB2 Information Center

For DB2 Version 9.5 topics, the DB2 Information Center URL is http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/

For DB2 Version 9 topics, the DB2 Information Center URL is http://publib.boulder.ibm.com/infocenter/db2luw/v9/

For DB2 Version 8 topics, go to the Version 8 Information Center URL at: http://publib.boulder.ibm.com/infocenter/db2luw/v8/

Displaying topics in your preferred language in the DB2 Information Center

The DB2 Information Center attempts to display topics in the language specified in your browser preferences. If a topic has not been translated into your preferred language, the DB2 Information Center displays the topic in English.

- To display topics in your preferred language in the Internet Explorer browser:
  1. In Internet Explorer, click the **Tools** —> **Internet Options** —> **Languages...** button. The Language Preferences window opens.
  2. Ensure your preferred language is specified as the first entry in the list of languages.
     - To add a new language to the list, click the **Add...** button.

       Note: Adding a language does not guarantee that the computer has the fonts required to display the topics in the preferred language.
     - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
  3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.
- To display topics in your preferred language in a Firefox or Mozilla browser:
  1. Select the button in the **Languages** section of the **Tools** —> **Options** —> **Advanced** dialog. The Languages panel is displayed in the Preferences window.
  2. Ensure your preferred language is specified as the first entry in the list of languages.
     - To add a new language to the list, click the **Add...** button to select a language from the Add Languages window.
     - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
  3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.

On some browser and operating system combinations, you might have to also change the regional settings of your operating system to the locale and language of your choice.

Updating the DB2 Information Center installed on your computer or intranet server

If you have installed the DB2 Information Center locally, you can obtain and install documentation updates from IBM.

Updating your locally-installed DB2 Information Center requires that you:

1. Stop the DB2 Information Center on your computer, and restart the Information Center in stand-alone mode. Running the Information Center in stand-alone mode prevents other users on your network from accessing the Information Center, and allows you to apply updates. Non-Administrative and Non-Root DB2 Information Centers always run in stand-alone mode. .
2. Use the Update feature to see what updates are available. If there are updates that you would like to install, you can use the Update feature to obtain and install them

   Note: If your environment requires installing the DB2 Information Center updates on a machine that is not connected to the internet, you have to mirror the update site to a local file system using a machine that is connected to the internet and has the DB2 Information Center installed. If many users on your network will be installing the documentation updates, you can reduce the time required for individuals to perform the updates by also mirroring the update site locally and creating a proxy for the update site.
   If update packages are available, use the Update feature to get the packages. However, the Update feature is only available in stand-alone mode.
3. Stop the stand-alone Information Center, and restart the DB2 Information Center on your computer.

Note: On Windows Vista, the commands listed below must be run as an administrator. To launch a command prompt or graphical tool with full administrator privileges, right-click on the shortcut and then select **Run as administrator**.

To update the DB2 Information Center installed on your computer or intranet server:

1. Stop the DB2 Information Center.
   - On Windows, click **Start** → **Control Panel** → **Administrative Tools** → **Services**. Then right-click on **DB2 Information Center** service and select **Stop**.
   - On Linux, enter the following command:
     ```
     /etc/init.d/db2icdv95 stop
     ```
2. Start the Information Center in stand-alone mode.
   - On Windows:
     a. Open a command window.
     b. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the <Program Files>\IBM\DB2 Information Center\Version 9.5 directory, where <Program Files> represents the location of the Program Files directory.
     c. Navigate from the installation directory to the doc\bin directory.
     d. Run the help_start.bat file:
        ```
        help_start.bat
        ```
   - On Linux:

a. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the /opt/ibm/db2ic/V9.5 directory.
b. Navigate from the installation directory to the doc/bin directory.
c. Run the help_start script:

```
help_start
```

The systems default Web browser launches to display the stand-alone Information Center.

3. Click the **Update** button (). On the right hand panel of the Information Center, click **Find Updates**. A list of updates for existing documentation displays.
4. To initiate the installation process, check the selections you want to install, then click **Install Updates**.
5. After the installation process has completed, click **Finish**.
6. Stop the stand-alone Information Center:
   - On Windows, navigate to the installation directory's doc\bin directory, and run the help_end.bat file:

     ```
     help_end.bat
     ```

     Note: The help_end batch file contains the commands required to safely terminate the processes that were started with the help_start batch file. Do not use `Ctrl-C` or any other method to terminate help_start.bat.
   - On Linux, navigate to the installation directory's doc/bin directory, and run the help_end script:

     ```
     help_end
     ```

     Note: The help_end script contains the commands required to safely terminate the processes that were started with the help_start script. Do not use any other method to terminate the help_start script.
7. Restart the DB2 Information Center.
   - On Windows, click **Start** → **Control Panel** → **Administrative Tools** → **Services**. Then right-click on **DB2 Information Center** service and select **Start**.
   - On Linux, enter the following command:

     ```
     /etc/init.d/db2icdv95 start
     ```

The updated DB2 Information Center displays the new and updated topics.

DB2 tutorials

The DB2 tutorials help you learn about various aspects of DB2 products. Lessons provide step-by-step instructions.

Before you begin

You can view the XHTML version of the tutorial from the Information Center at http://publib.boulder.ibm.com/infocenter/db2help/.

Some lessons use sample data or code. See the tutorial for a description of any prerequisites for its specific tasks.

DB2 tutorials

To view the tutorial, click on the title.

"pureXML™" in *pureXML Guide*
Set up a DB2 database to store XML data and to perform basic operations with the native XML data store.

"Visual Explain" in *Visual Explain Tutorial*
Analyze, optimize, and tune SQL statements for better performance using Visual Explain.

DB2 troubleshooting information

A wide variety of troubleshooting and problem determination information is available to assist you in using DB2 products.

DB2 documentation
Troubleshooting information can be found in the DB2 Troubleshooting Guide or the Support and Troubleshooting section of the DB2 Information Center. There you will find information on how to isolate and identify problems using DB2 diagnostic tools and utilities, solutions to some of the most common problems, and other advice on how to solve problems you might encounter with your DB2 products.

DB2 Technical Support Web site
Refer to the DB2 Technical Support Web site if you are experiencing problems and want help finding possible causes and solutions. The Technical Support site has links to the latest DB2 publications, TechNotes, Authorized Program Analysis Reports (APARs or bug fixes), fix packs, and other resources. You can search through this knowledge base to find possible solutions to your problems.

Access the DB2 Technical Support Web site at http://www.ibm.com/software/data/db2/udb/support.html

Terms and Conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal use: You may reproduce these Publications for your personal, non commercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these Publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the Publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Appendix B. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country/region 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/region 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.

This document may provide links or references to non-IBM Web sites and resources. IBM makes no representations, warranties, or other commitments whatsoever about any non-IBM Web sites or third-party resources that may be referenced, accessible from, or linked from this document. A link to a non-IBM Web site does not mean that IBM endorses the content or use of such Web site or

its owner. In addition, IBM is not a party to or responsible for any transactions you may enter into with third parties, even if you learn of such parties (or use a link to such parties) from an IBM site. Accordingly, you acknowledge and agree that IBM is not responsible for the availability of such external sites or resources, and is not responsible or liable for any content, services, products, or other materials on or available from those sites or resources. Any software provided by third parties is subject to the terms and conditions of the license that accompanies that software.

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 that has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
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 document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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 illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

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 are trademarks or registered trademarks of the International Business Machines Corporation in the United States, other countries, or both.

pureXML
Informix
DB2
Candle
System z
Parallel Sysplex
WebSphere
DB2 Connect
Redbooks
System i
IBM
zSeries
SQL/DS
S/390
HACMP
MVS
Approach
ibm.com
iSeries
Distributed Relational Database Architecture
ESCON
AIX
VTAM
i5/OS
Encina
OS/390
DB2 Universal Database
z/OS
CICS
RACF
Cloudscape
Lotus
Rational
DRDA
OS/400
Domino
NetView

The following terms are trademarks or registered trademarks of other companies

- Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
- Java, and and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.
- Microsoft, and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Printed in USA

SC23-5841-01

Spine information:

DB2 Connect Version 9.5

DB2 Connect User's Guide